Low Rank Tensor Methods in Galerkin-based Isogeometric Analysis

Angelos Mantzaflaris, Bert Jüttler, Boris Khoromskij, Ulrich Langer

Low Rank Tensor Methods in Galerkin-based Isogeometric Analysis

Angelos Mantzaflaris [a] Bert Jüttler [a] Boris N. Khoromskij [b]
Ulrich Langer [a]

[a] *Radon Institute for Computational and Applied Mathematics (RICAM), Austrian Academy of Sciences, Linz, Austria*

[b] *Max-Planck-Institute for Mathematics in the Sciences, Leipzig, Germany*

Abstract

The global (patch-wise) geometry map, which describes the computational domain, is a new feature in isogeometric analysis. This map has a global tensor structure, inherited from the parametric spline geometry representation. The use of this global structure in the discretization of partial differential equations may be regarded as a drawback at first glance, as opposed to the purely local nature of (high-order) classical finite elements. In this work we demonstrate that it is possible to exploit the regularity of this structure and to identify the great potential for the efficient implementation of isogeometric discretizations. First, we formulate tensor-product B-spline bases as well as the corresponding mass and stiffness matrices as tensors in order to reveal their intrinsic structure. Second, we derive an algorithm for the the separation of variables in the integrands arising in the discretization. This is possible by means of low rank approximation of the integral kernels. We arrive at a compact, separated representation of the integrals. The separated form implies an expression of Galerkin matrices as Kronecker products of matrix factors with small dimensions. This representation is very appealing, due to the reduction in both memory consumption and computation times. Our benchmarks, performed using the C++ library G+Smo, demonstrate that the use of tensor methods in isogeometric analysis possesses significant advantages.

Key words: isogeometric analysis, low rank approximation, stiffness matrix, matrix formation, tensor decomposition, Kronecker product, numerical quadrature

Email addresses: angelos.mantzaflaris@oeaw.ac.at (Angelos Mantzaflaris), bert.juettler@ricam.oeaw.at (Bert Jüttler), bokh@mis.mpg.de (Boris N. Khoromskij), ulrich.langer@ricam.oeaw.at (Ulrich Langer).

1 Introduction

Isogeometric analysis (IGA) [?,?] uses tensor-product B-spline bases for both the geometry description and the discretization of partial differential equations (PDEs). For a Galerkin-based approach, the 3D geometry is represented by tri-variate B-spline volumes. This significantly increases the complexity and, consequently, the computation times of matrix formation, which involves expensive multidimensional quadrature. Our aim is to apply tensor numerical methods, which are based on the idea of a low-rank *separable tensor decomposition* of integral kernels in order to obtain a fast generation technology of three-dimensional Galerkin matrices. Tensors generalize matrices to higher dimensions, and are the objects of interest when working with tensor-product B-splines in dimension three or higher.

In the present work, an automatic, input sensitive procedure is deduced that is able to capture the complexity of a given isogeometric domain and generate an adapted representation which is more efficient to manipulate in simulations. For example, if the input is a 3D cube, the computations will complete much faster than for a general freeform volume. Here the complexity of a domain is quantified by a rank parameter.

We represent the Galerkin matrices using the language of tensors. Then we employ tensor decomposition to derive a compact *Kronecker format* for these matrices which drastically reduces computation times for matrix formation. In particular, we prove a computation cost proportional to the geometric complexity of the input isogeometric domain which is linear in the size of the computed matrix. The overall error of the procedure is controlled by the threshold in the low rank tensor approximation. In this way, we reduce demanding multi-dimensional quadrature operations on tensor-product B-splines to inexpensive one-dimensional operations on univariate B-splines. Finally, our experiments demonstrate how one can use the Kronecker product format as a black-box matrix in matrix-vector evaluations in iterative solvers, leading to drastical reduction of memory requirements in the overall process.

1.1 Related work

There exist several lines of research dealing with the problem of increased computation costs in IGA. For the problem of matrix formation, we may refer to works exploring new rules with fewer quadrature points for numerical integration. These new rules are typically developed and computed numerically in 1D, and then they are extended in more dimensions by the use of tensor-product structure [?,?,?,?].

Moreover, there are works that assume that a 1D integration routine is available, and aim at efficiently applying it to more dimensions. Other lines of work in this direction include reduced quadrature [?], variationally consistent quadrature [?], isogeometric collocation methods [?,?,?,?,?] and adaptivity using adaptively refined meshes [?,?,?].

Another line of work is towards exploitation of the tensor-product structure of multivariate B-splines. This idea goes back to [?]. In particular, the technique of sum factorization employed in isogeometric analysis in [?], notably combined with weighted quadrature rules [?]. In the integration by interpolation and lookup approach, introduced

in [?], a lower degree is used to approximate the Jacobian determinant and reduce the complexity of the representation. Subsequently the resulting elementary integrals are pre-computed by the use of compact look-up tables. A main issue is that a small gain is observed for low degree (quadratic or cubic), which nevertheless are the most common degrees used in applications.

Finally, let us mention works related to the efficient manipulation of isogeometric data in solvers. These are based on the use of Kronecker product structure for preconditioning [?,?,?] as well as to the fast solution of the Sylvester equation [?,?].

Our main tool is the rank-structured tensor decomposition and the use of multilinear algebra. Traditional numerical approximations of PDEs in $\mathbb{R}^d$ are computationally tractable only for a moderate number of spacial variables due to the "curse of dimensionality", i.e. the fact that storage demands and complexity costs grow exponentially with the dimension d. The breaking through approach of data-sparse representation of d-variate functions and operators on large tensor-product grids is based on the principle of separation of variables. The low-parametric representations of high-dimensional data arrays employs the traditional canonical, Tucker and matrix product states (tensor train) tensor formats [?,?,?,?,?]. Literature surveys on the methods of multilinear algebra on rank-structured tensors can be found in [?,?,?,?].

The modern grid-based tensor methods for solving multidimensional PDEs employ the commonly used low-parametric rank-structured tensor formats, thus requiring merely linear in d storage costs $O(dn)$ for representation of function related tensors of size n^d, where n is usually associated with a large univariate grid-size, see survey papers [?,?]. Efficient tensor methods for the treatment of function related tensors (obtained by sampling a continuous function on the large tensor grid) via canonical and Tucker formats were described in [?,?].

The method of quantized tensor approximation (QTT) is proven to provide a logarithmic data-compression for a wide class of discrete functions and operators [?]. It enables to discretize and solve multi-dimensional steady-state and dynamical problems with a logarithmic complexity $O(d \log n)$ in the volume size of the computational tensor-product grid. The decoupling of multivariate functions using tensor decomposition is further analyzed in [?].

1.2 Contributions and outline

Our work extends [?], where the low-rank approximation in the 2D case is treated by means of singular value decomposition (SVD). Two basic properties of isogeometric analysis are important for the method. First, the tensor-product B-spline functions have the property of being separable, i.e., they are the product of univariate basis functions. Second, the kernels appearing in isogeometric integrals typically involve coefficients of a PDE and the partial derivatives of the geometry map, all defined globally on a patch representing the domain. These two properties pave the way for the use of numerical tensor calculus.

Our approach suggests to delay multi-dimensional operations as long as possible. This is possible due to the tensor-product nature of the B-spline basis and the global

geometry map. In fact, our final aim is to refrain from multi-dimensional operations completely, as far as the matrix assembly and the solution process is concerned.

We focus on the task of Galerkin matrix formation and we consider the mass and the stiffness matrix as model cases. For a tensor-product patch with $O(n)$ degrees of freedom per parametric dimension, the number of non-zero entries in these matrices is $O(n^d p^d)$, where d denotes the spatial dimension (e.g. $d = 2$ or 3) and p is the spline degree used for each dimension. Since the time complexity of the method is bounded by the size of the output, this is a lower bound for the asymptotic computation time of the problem. We present a procedure that achieves (quasi-optimal) complexity $O(Rdn^d p^d)$, where R is a rank parameter related to the *geometric complexity* of the input domain.

Concerning the implementation, the method requires 1D interpolation and matrix formation routines, a tensor decomposition routine and the Kronecker product operation. Therefore, the method can be used on top of existing isogeometric procedures, and does not require a change of paradigm. For instance, any known quadrature rule for univariate splines can be employed, e.g. [**?**,**?**].

The paper is organized as follows. In Section 2 we briefly describe the isogeometric paradigm and we focus on the specific integrals that arise in the process. Section 3 is devoted to a short introduction to multilinear algebra dealing with such rank-structured representations. In Section 4 we establish tensor notation for B-splines, and we define the mass and stiffness tensors in Section 5. Our main algorithm is summarized in Section 6, where we also provide a complexity analysis. We conclude with experiments and brief conclusions in Sections 7 and 8, respectively.

2 Isogeometric discretization

A Galerkin-based isogeometric simulation is performed on a parameterized physical domain Ω. The domain is parameterized by a global geometry mapping $G : \hat{\Omega} \to \Omega$, where the parameter domain $\hat{\Omega}$ is an axis-aligned box in $\mathbb{R}^d$. More generally, one might consider a collection of such boxes for a multi-patch domain parameterization [**?**,**?**]. In order to keep the presentation simple and since this suffices to describe the main ideas we restrict ourselves to only one patch.

Any point $\boldsymbol{x} \in \Omega$ in the physical domain is the image of a point $\boldsymbol{\xi} \in \hat{\Omega}$ in the parameter domain,

$$\boldsymbol{x} = G(\boldsymbol{\xi}) = \sum_{\boldsymbol{i} \in \mathcal{I}} \boldsymbol{d_i}\, \beta_{\boldsymbol{i}}(\boldsymbol{\xi})\,. \tag{1}$$

The geometry mapping is represented using basis functions $\beta_{\boldsymbol{i}}$ (which are typically NURBS functions), which are multiplied by control points $\boldsymbol{d_i}$. The latter ones possess an intuitive geometric meaning and are well-established tools in geometric design [**?**].

The isogeometric simulation takes advantage of the given parameterization of the domain Ω. In particular, the isogeometric discretization space is defined as

$$V_h = \operatorname{span}\{\beta_{\boldsymbol{i}} \circ G^{-1} \;:\; \boldsymbol{i} \in \mathcal{I}\}. \tag{2}$$

The functions in the discretization space V_h are linear combinations of the basis functions,

$$u_h = \sum_{\boldsymbol{i} \in \mathcal{I}} u_{\boldsymbol{i}} \, (\beta_{\boldsymbol{i}} \circ G^{-1}) \,, \tag{3}$$

with certain coefficients $\boldsymbol{u} = (u_{\boldsymbol{i}})_{\boldsymbol{i} \in \mathcal{I}}$, where $\mathcal{I}$ is a finite (multi-)index set.

Here, for any function (e.g. u) which is defined on the physical domain Ω, we use a Greek character (e.g. ϕ) to denote its pull-back $\phi = u \circ G$.

The finite-dimensional space V_h is now used for the Galerkin discretization of the variational (weak) formulation of some boundary value problem for a linear elliptic PDE that can be written in the form: find

$$u \in V \text{ such that } \; a(u, v) = \ell(v), \quad \forall v \in V \,. \tag{4}$$

In the case of a scalar second-order PDE and natural boundary conditions, V is nothing but the Sobolev space $H^1(\Omega)$. In the presence of Dirichlet boundary conditions imposed on some part Γ_D of the boundary Γ of the physical domain Ω, u must satisfy this Dirichlet boundary conditions, whereas the test functions v have to vanish on Γ_D. In the following, it is obviously enough to consider the case of natural boundary conditions.

It should be noted that the isogeometric discretization is often derived from a representation (1) of the geometry mapping, which is obtained by applying several refinement steps (usually h-refinement / knot insertion) to the original representation of the mapping, in order to generate a sufficiently fine space for the simulation.

After applying the Galerkin method to the variational form (4), we arrive at the discretized problem, which consists in finding

$$u_h \in V_h \subset V \text{ such that } \; a(u_h, v_h) = \ell(v_h), \quad \forall v_h \in V_h \,. \tag{5}$$

We refer to classical text books on the finite element method like [**?**,**?**,**?**] for a detailed presentation of Galerkin method and the corresponding numerical analysis.

Considering the test functions $v_h = \beta_{\boldsymbol{i}} \circ G^{-1}$, $\boldsymbol{i} \in \mathcal{I}$, the variational form (5) leads to the linear system of algebraic equations

$$S\boldsymbol{u} = \boldsymbol{b},$$

which characterizes the isogeometric solution u_h with coefficients $\boldsymbol{u}$, cf. (3).

In the remainder of the paper, we focus on typical ingredients of the bilinear form. Let

$$\phi_h = u_h \circ G \quad \text{and} \quad \psi_h = v_h \circ G.$$

We consider the mass term

$$a_M(u_h, v_h) = \int_{\hat{\Omega}} \phi_h \, \psi_h \, \omega \, \mathrm{d}\boldsymbol{\xi}, \quad \omega = |\det \nabla G|, \tag{6}$$

the stiffness term

$$a_S(u_h, v_h) = \int_{\hat{\Omega}} \nabla\phi_h{}^\top \, K \, \nabla\psi_h \, \mathrm{d}\boldsymbol{\xi}, \quad K = [\kappa_{\ell,m}] = \frac{(\nabla G)_c^\top \, (\nabla G)_c}{|\det \nabla G|}, \tag{7}$$

and the advection term

$$a_A(u_h, v_h) = \int_{\hat{\Omega}} \left(\nabla\phi_h{}^\top \boldsymbol{\varrho}\right) \, \psi_h \, \mathrm{d}\boldsymbol{\xi}, \quad \boldsymbol{\varrho} = (\nabla G)_c^\top \, (\boldsymbol{r} \circ G). \tag{8}$$

Their definition involves the Jacobian matrix ∇G, its (transposed) cofactor matrix $(\nabla G)_c^\top = (\nabla G)^{-1} \det \nabla G$, and the advection vector field $\boldsymbol{r}$. The elements of the system matrix S take the form

$$S_{\boldsymbol{ij}} = a(\beta_{\boldsymbol{i}}, \beta_{\boldsymbol{j}}) \,, \tag{9}$$

where the bilinear form a is essentially a linear combination of the terms in (6-8). For example, the Neumann problem for the PDE $-\Delta u + r^\top \nabla u + u = f$ leads to the bilinear form $a = a_S + a_A + a_M$. Other terms that appear in the discretization of variational forms of PDEs yield similar expressions. In particular, it is easy to consider additional coefficients in the diffusion and reaction terms of the PDE that we can incorporate in K and ω, respectively. In order to keep the presentation simple, we discuss neither terms arising from boundary conditions nor the right-hand side of the system.

3 Tensor calculus and formats

We recall fundamentals of tensor algebra and numerical tensor calculus. Additional information and further details are provided in the rich literature on this topic, see e.g. [?,?,?,?] and references therein.

3.1 Tensors

Given a non-negative integer $d \in \mathbb{Z}_{\geq 0}$ and a vector $\boldsymbol{n} = (n_1, \ldots, n_d) \in \mathbb{Z}_+^n$ with positive integer elements, we consider the index set

$$\mathcal{I} = I_1 \times \ldots \times I_d = \bigtimes_{k=1}^{d} I_k, \quad I_k = \{1, \ldots, n_k\}.$$

A (real) $\boldsymbol{n}$-tensor

$$\mathbf{T} = [t_{\boldsymbol{i}}]_{\boldsymbol{i} \in \mathcal{I}} \in \mathbb{W}_{\boldsymbol{n}} = \mathbb{R}^{\mathcal{I}}$$

of *order* d and *dimension* $\boldsymbol{n}$ is an array consisting of real elements $t_{\boldsymbol{i}} = t_{i_1,\ldots,i_d} \in \mathbb{R}$ with indices $\boldsymbol{i} = (i_1, \ldots, i_d) \in \mathcal{I}$, i.e., $i_k \in I_k$, $k = 1, \ldots, d$. For example, the elements of a $(3, 3, 3)$-tensor are visualized in Fig. 1. The number of elements

$$\pi(\boldsymbol{n}) = n_1 \cdots n_d$$

is called the *size* of the tensor. The set of $\boldsymbol{n}$-tensors forms the linear space $\mathbb{W}_{\boldsymbol{n}}$, equipped with the Euclidean scalar product.

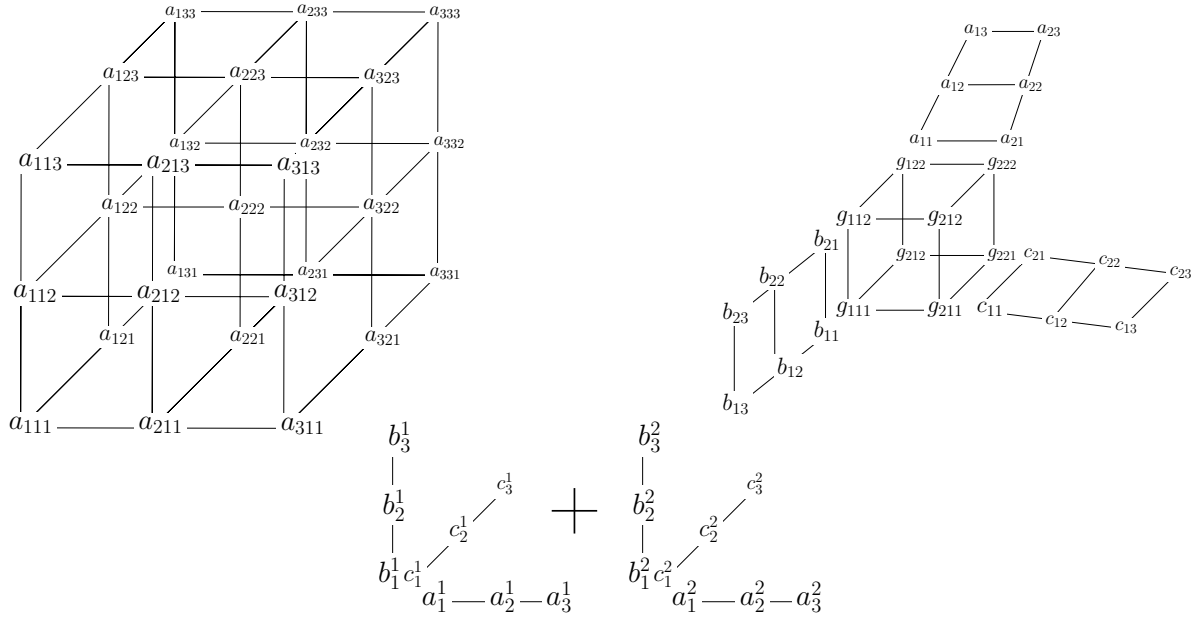


Fig. 1. A 3-tensor $\mathbf{A} = [a_{i_1,i_2,i_3}]$ with $\boldsymbol{i} = (i_1, i_2, i_3) \in \{1, 2, 3\}^3$. From left to right: full format, Tucker format with rank $(2, 2, 2)$, and canonical format of rank 2 at the bottom.

First and second order tensors, where the dimension takes the form $\boldsymbol{n} = (n_1)$ and $\boldsymbol{n} = (n_1, n_2)$, respectively, play a special role. Any (n_1)-tensor $\mathbf{v}$ is simply a *vector* and it will be denoted by a lowercase bold character. Any (n_1, n_2)-tensor A is a *matrix* and will be denoted by an uppercase italic character. Additionally it is possible to see real *scalars* as tensors of order $d = 0$.

The *vectorization* of a tensor, $\mathrm{vec}(\mathbf{T})$, results in a vector which is obtained simply by reshaping the multi-index $\boldsymbol{i}$ to a *single* index using a lexicographic ordering in $\mathcal{I}$. A *matricization* $\mathrm{mat}(\mathbf{T})$ of a tensor is matrix obtained by splitting the indices $i_1, \ldots, i_d$ into two disjoint subsets, which are ordered lexicographically and used to assign every tensor entry to matrix row and column.

In order to facilitate the analysis of the computational complexity we will often consider $\boldsymbol{n}$-tensors with $n_i = n$; these will be denoted as nd-tensors. The storage requirements of a nd-tensor grow exponentially with d since $\dim(\mathbb{W}_{\boldsymbol{n}}) = n^d$. This fact, which is sometimes called the "curse of dimensionality", makes it virtually impossible to use the traditional numerical methods, since they are characterized by linear complexity in the discrete problem size, even for moderate values of the dimension d. Instead it is preferable to employ memory-efficient representations, which are based on tensor-products.

3.2 Tensor product and tensor formats

The *tensor product* of an $\boldsymbol{n}$-tensor $\mathbf{T} = [t_{\boldsymbol{i}}]$ and an $\bar{\boldsymbol{n}}$-tensor $\bar{\mathbf{T}} = [\bar{t}_{\bar{\boldsymbol{i}}}]$ is the $(\boldsymbol{n}, \bar{\boldsymbol{n}})$-tensor $\mathbf{P}$ that is formed by the products of all pairs of elements,

$$\mathbf{P} = \mathbf{T} \otimes \bar{\mathbf{T}} = [p_{\boldsymbol{i},\bar{\boldsymbol{i}}}]_{(\boldsymbol{i},\bar{\boldsymbol{i}}) \in \mathcal{I} \times \bar{\mathcal{I}}}, \quad p_{\boldsymbol{i},\bar{\boldsymbol{i}}} = p_{i_1,\ldots,i_d,\bar{i}_1,\ldots,\bar{i}_{\bar{d}}} = t_{\boldsymbol{i}} t_{\bar{\boldsymbol{i}}} = t_{i_1,\ldots,i_d} \bar{t}_{\bar{i}_1,\ldots,\bar{i}_{\bar{d}}} \,. \tag{10}$$

Moreover, the tensor product

$$\mathbf{Q} = \bigotimes_{k=1}^{d} \mathbf{v}^{(k)} \quad \text{with elements} \quad q_{i_1,\ldots,i_d} = \prod_{k=1}^{d} v_{i_k}^{(k)} \tag{11}$$

of d vectors $\mathbf{v}^{(k)}$ of dimensions n_k, where $k = 1, \ldots, d$, is an $\boldsymbol{n} = (n_1, \ldots, n_d)$-tensor, is referred to as a *rank-1 tensor*. Note that the storage costs of this tensor scales linearly in the order d.

Multiplication by scalars, which are seen as tensors of order 0, is a special case of the tensor product, where one usually omits the $\otimes$ sign. It satisfies the identity

$$\left(\prod_{j=1}^{k} s^{(j)}\right) \left(\bigotimes_{j=1}^{k} \mathbf{T}^{(j)}\right) = \bigotimes_{j=1}^{k} \left(s^{(j)} \mathbf{T}^{(j)}\right). \tag{12}$$

Using the tensor product leads to memory-efficient representations of tensors. In our paper we use the two main formats (or representations) which are common in the literature:

- The *R-term canonical representation* of a tensor takes the form

$$\mathbf{T} = [t_{\boldsymbol{i}}] = \sum_{r=1}^{R} \bigotimes_{k=1}^{d} \mathbf{v}^{(k,r)}, \quad \text{i.e.,} \quad t_{\boldsymbol{i}} = t_{i_1,\ldots,i_d} = \sum_{r=1}^{R} \prod_{k=1}^{d} v_{i_k}^{(k,r)}, \tag{13}$$

  and is parameterized by Rd vectors $\mathbf{v}^{(k,r)} \in \mathbb{R}^{n_k}$. Clearly, any tensor admits a canonical representation for a sufficiently large value of R. The smallest integer R such that a given tensor $\mathbf{T}$ admits such a representation is called its (canonical) rank R. If an nd-tensor has rank R or lower, then its storage cost is bounded by Rnd.

  An important special case is $R = 1$, where the representation takes the form (11). Tensors admitting a rank-1 representation are said to be *separable*.

  In the case of matrices, the singular value decomposition (SVD) creates a canonical representation, where the rank R is equal to the number of non-zero singular values and the vectors $\mathbf{v}^{(k,r)}$ are the left and right singular vectors scaled by the square roots of the singular values.
- A more general representation is the *Tucker format*

$$\mathbf{T} = [t_{\boldsymbol{i}}] = \sum_{r_1=1}^{\rho_1} \cdots \sum_{r_d=1}^{\rho_d} c_{r_1,\ldots,r_d} \bigotimes_{k=1}^{d} \mathbf{v}^{(k,r_k)}, \text{i.e., } t_{\boldsymbol{i}} = t_{i_1,\ldots,i_d} = \sum_{r_1=1}^{\rho_1} \cdots \sum_{r_d=1}^{\rho_d} c_{r_1,\ldots,r_d} \prod_{k=1}^{d} v_{i_k}^{(k,r_k)}. \tag{14}$$

  which is defined by specifying the, so called, multi-rank $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_d)$, the $\rho_1 + \cdots + \rho_d$ vectors $\mathbf{v}^{(k,r_k)} \in \mathbb{R}^{n_k}$ and the core tensor $\mathbf{C} = [c_{\boldsymbol{r}}]$ of order d and dimension $\boldsymbol{\rho}$. The storage cost is bounded by $d\rho n + \rho^d$ with $\rho = \max_k \rho_k$. This format becomes identical to the canonical format when choosing $\rho_k = R$ and considering a diagonal core tensor $\mathbf{C}$.

In the remainder of the paper we will focus mostly on the canonical representation. However, a representation in the Tucker format can be converted into a canonical representation, and vice versa. Notice that the maximal canonical rank of a tensor is

bounded by $\pi(\boldsymbol{n})/\max\{n_k\}$, while the canonical rank of the Tucker tensor does not exceed $\pi(\boldsymbol{\rho})/\max\{\rho_k\}$.

3.3 Binary operations on tensors

Besides the tensor product, numerous product operations for tensors are available. They can be defined by applying summation to certain subsets of the elements of the tensor product (10) or by performing a matricization (in the case of matrices). We will use the operations which are listed in Table 1.

Table 1
Selected binary operations for two tensors $\mathbf{T} = [t_{\boldsymbol{i}}] \in \mathbb{W}_{\boldsymbol{n}}$ and $\bar{\mathbf{T}} = [\bar{t}_{\bar{\boldsymbol{i}}}] \in \mathbb{W}_{\bar{\boldsymbol{n}}}$.

Name	symbol	order of result	obtained from the tensor product (10) by	compatibility requires
contracted	$\cdot$	$d + \bar{d} - 2$	summation over indices with $i_1 = \bar{i}_1$	$n_1 = \bar{n}_1$
Frobenius	$:$	$d - \bar{d}$	summation over indices with $i_k = \bar{i}_k$, $k = 1, \ldots, \bar{d}$	$d \geq \bar{d}$ and $n_k = \bar{n}_k$ for $k = 1, \ldots, \bar{d}$.
Kronecker	$\otimes$	2	matricization with respect to $(i_1, \bar{i}_1)$ and $(i_2, \bar{i}_2)$	$d = \bar{d} = 2$

These operations are restricted to compatible pairs of tensors, and the Kronecker product is defined for pairs of matrices only. Sometimes it will be necessary to consider tensors with permuted indices to achieve compatibility. We will use the tilde to indicate the application of an index permutation σ. More precisely,

$$\mathbf{T}^{\sim} = [t_{\sigma(\boldsymbol{i})}]$$

is a tensor obtained from $\mathbf{T} = [t_{\boldsymbol{i}}]$ by suitably permuting indices, and $\cong$ denotes equality of tensors up to index permutations, i.e.,

$$\mathbf{T} \cong \bar{\mathbf{T}} \quad \Leftrightarrow \quad \mathbf{T}^{\sim} = \bar{\mathbf{T}}.$$

The choice of the permutation σ should always be clear from the context.

Some remarks about these product operations are in order:

- The contracted product

$$\mathbf{P} = \mathbf{T} \cdot \bar{\mathbf{T}} = [t_{\boldsymbol{i}}] \cdot [\bar{t}_{\bar{\boldsymbol{i}}}] \quad \text{with elements} \quad p_{i_2,\ldots,i_d,\bar{i}_2,\ldots,\bar{i}_{\bar{d}}} = \sum_{i_1=1}^{n_1} t_{i_1,i_2,\ldots,i_d} \bar{t}_{i_1,\bar{i}_2,\ldots,\bar{i}_{\bar{d}}} \,.$$

  generates a tensor of order $d + \bar{d} - 2$. It is possible to perform summations over other index pairs than the first ones (i.e., $i_j = \bar{i}_k$ for general j, k). Again, this will be expressed as $\mathbf{T}^{\sim} \cdot \bar{\mathbf{T}}^{\sim}$. In particular, this covers the case of matrix multiplication,

$$AB = A^{\sim} \cdot B \quad \text{with} \quad A^{\sim} = A^{\top},$$

where we used the transposition operation to comply with the convention that the summation considers the first indices of both tensors.

- The Frobenius product

$$\mathbf{P} = \mathbf{T} : \bar{\mathbf{T}} = [t_{\boldsymbol{i}}] : [\bar{t}_{\bar{\boldsymbol{i}}}] \quad \text{with elements} \quad p_{i_{\bar{d}+1},\ldots,i_d} = \sum_{i_1,\ldots,i_{\bar{d}}} t_{i_1,\ldots,i_{\bar{d}},i_{\bar{d}+1},\ldots,i_d} \bar{t}_{i_1,\ldots,i_{\bar{d}}} \ .$$

performs summation with respect to as many index pairs as possible, where this is determined by the order of the second factor, which is assumed to not exceed the order of the first one. Again it is possible to consider other sequences of pairs of indices.

This product gives a scalar number if both factors have the same order and dimension. It thus defines an inner product and the norm $\|\cdot\|_F$ on $\mathbb{W}_{\boldsymbol{n}}$ in this case.

The Frobenius product of separable tensors of the same dimensions satisfies the useful identity

$$\left(\bigotimes_{i=1}^{d} \mathbf{v}^{(i)}\right) : \left(\bigotimes_{i=1}^{d} \bar{\mathbf{v}}^{(i)}\right) = \prod_{i=1}^{d} (\mathbf{v}^{(i)} \cdot \bar{\mathbf{v}}^{(i)}). \tag{15}$$

- The Kronecker product

$$A \otimes \bar{A} = \begin{bmatrix} a_{11}\bar{A} & \cdots & a_{1,n_2}\bar{A} \\ \vdots & \ddots & \vdots \\ a_{n_1,1}\bar{A} & \cdots & a_{n_1,n_2}\bar{A} \end{bmatrix},$$

is only defined for pairs of matrices. It is a particular matricization of the tensor product of the two matrices A and $\bar{A}$. The Kronecker product is a non-commutative operation. It relates the tensor product operation to standard matrix operations. In particular, the Kronecker product of two matrices (2-tensors) is a specific matricization of their tensor product (which is 4-tensor). More precisely, if $A = [a_{ij}]$ and $B = [b_{nm}]$, we obtain

$$\mathrm{mat}(A \otimes B) = A \otimes B \ , \tag{16}$$

where the matricization on the left-hand side is defined by collapsing the indices i, n to a single row index and the indices j, m to a single column index.

3.4 Algorithms for tensor decomposition

The rank-structured tensors with controllable complexity (i.e., the set of tensors in canonical or Tucker formats with bounded rank) form a manifold $\mathbb{S} \subset \mathbb{W}_{\boldsymbol{n}}$ [**?**]. In order to reduce the complexity of computations with tensors, we need to find approximate representations of general tensors in this manifold, by performing a projection into $\mathbb{S}$. This projection takes the form of the *tensor truncation operator*

$$T_{\mathbb{S}} : \mathbb{W}_{\boldsymbol{n}} \to \mathbb{S} : \qquad T_{\mathbb{S}}\mathbf{A} = \operatorname*{argmin}_{\mathbf{U}\in\mathbb{S}} \|\mathbf{A} - \mathbf{U}\|, \tag{17}$$

and leads to a non-linear optimization problem, which is usually solved only approximately.

For projecting a tensor to the *canonical rank-R format*, one frequently used approach is the alternating least-squares (ALS) method for non-linear approximation. This is an iterative optimization algorithm which minimizes the squared Frobenius norm

$$f\left(V^{(1)},\dots,V^{(d)}\right) = \left\| \mathbf{A} - \sum_{r=1}^{R} \bigotimes_{k=1}^{d} \mathbf{v}^{(k,r)} \right\|_F^2 .$$

Here $V^{(k)} \in \mathbb{R}^{n_k \times R}$ is the matrix with columns $\mathbf{v}^{(k,r)}$. We start from some initial guess for $V^{(k)}$, $k = 1,\dots,d$. One iteration of the ALS algorithm consists in successively solving d least-square problems

$$\begin{aligned} V^{(1)'} &= \operatorname*{argmin}_{U \in \mathbb{R}^{n_1 \times R}} f\left(U, V^{(2)},\dots,V^{(d)}\right) \\ &\vdots \\ V^{(d)'} &= \operatorname*{argmin}_{U \in \mathbb{R}^{n_d \times R}} f\left(V^{(1)'},\dots,V^{(d-1)'},U\right) . \end{aligned}$$

The process is repeated until a certain tolerance is reached, or until no improvement is observed. Several techniques have been developed to obtain a robust ALS iteration, including choice of initial solutions, imposing orthogonality constraints, incorporating line search, etc. [**?**].

A quasi-optimal rank-$\boldsymbol{\rho}$ approximation in the *Tucker format* is provided by the higher-order SVD (HOSVD) algorithm [**?**]. This algorithm is based on tensor unfoldings. In particular, we can transform a tensor $\mathbf{A}$ into a matrix by selecting a specific index k as the row index, and expanding the remaining ones to the column index. This matrix, which has dimension $n_k \times \prod_{j\neq k} n_j$ is the called the *k-unfolding* $A^{(k)}$. The first step of the HOSVD algorithm is to compute the SVD of all unfoldings, i.e.,

$$A^{(k)} = U^{(k)}\,\Sigma^{(k)}\,V^{(k)^T} \quad k = 1,\dots,d\,.$$

The skeleton vectors of the Tucker format in direction k are set to be the columns of $U^{(k)}$, $k = 1,\dots,d$. The second step produces the core tensor

$$\mathbf{C} = U^{(1)} \cdots U^{(d)} \cdot \mathbf{A}$$

where the computations are performed from right to left and the contracted product with $U^{(k)}$ is taken with respect to the $k-th$ index of $\mathbf{A}$. This completes the algorithm. Using a tolerance $\varepsilon > 0$ for truncating the SVDs in the first step yields an $\varepsilon-$approximation of $\mathbf{A}$ in the Tucker format.

The reduced HOSVD algorithm efficiently performs tensor truncation to the *canonical representation* for moderate dimensions, cf. [**?**, Th. 2.5]. For sufficiently smooth integral kernels, one obtains a canonical representation of rank bounded by $R = \rho^{d-1} = O(|\log\varepsilon|^{d-1}\log^{d-1} n)$, where ε is the error in the Tucker approximation of a (function-related) tensor. This bound can be proven for the class of tensors generated by analytic

functions with possible point singularities, see [?,?,?]. A further reduction of the canonical rank can be obtained by invoking the ALS algorithm.

4 Tensor functions

We proceed from tensors to functions with tensor-product structure. Tensor-product B-splines naturally fall into this category. Interestingly, there is a strong link between rank-1 tensors and multivariate functions which are products of univariate functions.

4.1 Tensor-product B-splines

We consider d possibly different univariate spline spaces $\mathcal{S}^{p_k}_{\boldsymbol{\tau}_k}$ with variables $\xi^{(k)}$, $k = 1, \ldots, d$. Each space is defined by a knot-vector $\boldsymbol{\tau}_k$ and a degree p_k. The standard basis of $\mathcal{S}^{p_k}_{\boldsymbol{\tau}_k}$ consists of the B-splines of degree p_k, which are associated with the knots $\boldsymbol{\tau}_k$, see [?]. We collect the basis functions in the vector

$$\boldsymbol{\beta}^{(k)}(\xi^{(k)}) = \left[\beta^{(k)}_{i_k}(\xi^{(k)})\right] \quad , \quad \xi^{(k)} \in [a_k, b_k]\,,\ k = 1 \ldots d\,,$$

which has the index set I_k possessing $n_k = \#\boldsymbol{\tau}_k - p_k - 1$ elements. More precisely, we obtain a vector (i.e., an element of $\mathbb{R}^{n_k}$) for any given value of the argument $\xi^{(k)}$. For any such value, at most $p_k + 1$ elements of the vector take non-zero values.

Given a coefficient vector $\mathbf{c}^{(k)}$, a spline function $f_k \in \mathcal{S}^{p_k}_{\boldsymbol{\tau}_k}$ can be represented using the inner product,

$$f_k(\xi^{(k)}) = \mathbf{c}^{(k)} \cdot \boldsymbol{\beta}^{(k)}(\xi^{(k)}).$$

We will omit the argument $\xi^{(k)}$ if no confusion can arise.

The $d-$variate tensor-product spline space $\mathcal{S}^{p_1}_{\boldsymbol{\tau}_1} \otimes \cdots \otimes \mathcal{S}^{p_d}_{\boldsymbol{\tau}_d}$ is the space of piecewise polynomial functions with variables $\boldsymbol{\xi} = (\xi^{(1)}, \ldots, \xi^{(d)})$ which is spanned by the tensor-product B-splines. These multivariate spline functions of degree $\boldsymbol{p} = (p_1, \ldots, p_d)$ are the elements of the separable $\boldsymbol{n}$-tensor

$$\mathbf{B}(\boldsymbol{\xi}) = \bigotimes_{k=1}^{d} \boldsymbol{\beta}^{(k)}(\xi^{(k)}) = [\beta_{\boldsymbol{i}}(\boldsymbol{\xi})] = \left[\prod_{k=1}^{d} \beta^{(k)}_{i_k}(\xi^{(k)})\right] \quad , \quad \boldsymbol{\xi} \in \hat{\Omega}. \tag{18}$$

More precisely, we obtain a tensor for any given value of the argument $\boldsymbol{\xi}$. Consequently, the tensor-product $\mathbf{B}$ is a function with values in $\mathbb{W}_{\boldsymbol{n}}$. For any given value $\boldsymbol{\xi}$, at most $\pi(\boldsymbol{p} + \mathbf{1})$ tensor-product B-splines take non-zero values.

Finally, we define the *gradient*

$$\nabla \otimes \mathbf{B}(\boldsymbol{\xi}) = \left[\frac{\partial}{\partial \xi^{(k)}} \beta_{i_1,\ldots,i_d}(\boldsymbol{\xi})\right]$$

of the tensor-product basis, which is a tensor of dimension $(d, n_1, \ldots, n_d)$.

4.2 Tensor-product spline functions

Each spline function $f \in \mathcal{S}^{p_1}_{\boldsymbol{\tau}_1} \otimes \cdots \otimes \mathcal{S}^{p_d}_{\boldsymbol{\tau}_d}$ depends on the d arguments $\boldsymbol{\xi} = (\xi^{(1)}, \ldots, \xi^{(d)})$ and possesses a unique representation

$$f(\boldsymbol{\xi}) = \mathbf{C} : \mathbf{B}(\boldsymbol{\xi}) = \sum_{\boldsymbol{i} \in \mathcal{I}} c_{\boldsymbol{i}}\, \beta_{\boldsymbol{i}}(\boldsymbol{\xi}), \tag{19}$$

where the coefficients form a tensor $\mathbf{C} \in \mathbb{W}_{(\boldsymbol{m},\boldsymbol{n})}$. Three types of spline functions will used in our paper:

(1) *Scalar-valued* spline functions are obtained by specifying an empty first part $\boldsymbol{m}$ of the dimension, thus $\mathbf{C} \in \mathbb{W}_{\boldsymbol{n}}$.

(2) For *vector-valued* spline functions, the first part of the dimension $\boldsymbol{m}$ is equal to the dimension d, thus $\mathbf{C} \in \mathbb{W}_{d,\boldsymbol{n}}$. The geometry mapping G, which has been introduced in (1), belongs to this class if it is a polynomial one. The elements of $\mathbf{B}$ are the basis functions $\beta_{\boldsymbol{i}}$, and the associated control points $\boldsymbol{d}_{\boldsymbol{i}}$ correspond to fibers (i.e., 1-slices) of the coefficient tensor $\mathbf{C}$.

Rational geometry mappings can be represented using homogeneous coordinates where $\mathbf{C} \in \mathbb{W}_{d+1,\boldsymbol{n}}$. For simplicity we restrict the presentation to the polynomial case but note that all results can be generalized to the more general case, cf. [**?**].

(3) *Matrix-valued* spline functions are obtained by considering dimensions with $\boldsymbol{m} = (d, d)$, thus $\mathbf{C} \in \mathbb{W}_{d,d,\boldsymbol{n}}$.

4.3 Rank-R spline functions

Owing to (15), a scalar-valued tensor-product spline function (19), which is defined by a rank$-R$ coefficient tensor

$$\mathbf{C} = \sum_{r=1}^{R} \bigotimes_{k=1}^{d} \mathbf{c}_r^{(k)},$$

is a sum of R products

$$f(\boldsymbol{\xi}) = \sum_{r=1}^{R} \prod_{k=1}^{d} f_r^{(k)}(\xi^{(k)}) \tag{20}$$

of the univariate spline functions

$$f_r^{(k)}(\xi^{(k)}) = \mathbf{c}_r^{(k)} \cdot \boldsymbol{\beta}^{(k)}(\xi^{(k)})\,.$$

We refer to any function f of the form (20) as a *rank-R function*, also if the univariate factors are not spline functions.

This observation can be carried over to matrix-valued spline functions

$$F(\boldsymbol{\xi}) = \mathbf{C} : \mathbf{B}(\boldsymbol{\xi}) = \sum_{\boldsymbol{i} \in \mathcal{I}} \begin{bmatrix} c_{1,1,\boldsymbol{i}} & \cdots & c_{1,d,\boldsymbol{i}} \\ \vdots & \ddots & \vdots \\ c_{d,1,\boldsymbol{i}} & \cdots & c_{d,d,\boldsymbol{i}} \end{bmatrix} \beta_{\boldsymbol{i}}(\boldsymbol{\xi})\,,$$

where one considers coefficient tensors $\mathbf{C} \in \mathbb{W}_{d,d,\boldsymbol{n}}$ with $\mathbf{C} = [c_{\ell,m,\boldsymbol{i}}]$ having the property that the slices $\mathbf{C}_{\ell,m}$ obtained by fixing the first two indices have rank R. These slices admit a representation

$$\mathbf{C}_{\ell,m} = \sum_{r=1}^{R} \bigotimes_{k=1}^{d} \mathbf{c}_{\ell,m,r}^{(k)}.$$

The elements of the matrix-valued function $F = [\phi_{\ell,m}]_{\ell,m=1,\dots,d}$ are sums of R products

$$\phi_{\ell,m}(\boldsymbol{\xi}) = \sum_{r=1}^{R} \prod_{k=1}^{d} \phi_{\ell,m,r}^{(k)}(\xi^{(k)}) \tag{21}$$

of univariate spline functions

$$\phi_{\ell,m,r}^{(k)}(\xi^{(k)}) = \mathbf{c}_{\ell,m,r}^{(k)} \cdot \boldsymbol{\beta}^{(k)}(\xi^{(k)})\,.$$

These functions form *factor matrices*

$$F_r^{(k)}(\xi^{(k)}) = [\phi_{\ell,m,r}^{(k)}(\xi^{(k)})]_{\ell,m=1,\dots,d} = \mathbf{C}_r^{(k)\sim} \cdot \boldsymbol{\beta}^{(k)}(\xi^{(k)})$$

with coefficient tensors $\mathbf{C}_r^{(k)} = [\mathbf{c}_{\ell,m,r}^{(k)}]_{\ell,m=1,\dots,d}$. The index permutation rotates the indices such that the last one takes the first place. Again we refer to a function F of the form (21) as *matrix-valued rank-R function*, also if the univariate matrix-valued factors are not spline functions.

Note that each of the d^2 elements of the matrix-valued function $F(\boldsymbol{\xi})$ may have a different rank. Thus one may consider a $d \times d$ matrix of coordinate-wise ranks $R_{\ell,m}$ and define the overall rank of $F(\boldsymbol{\xi})$ as $R = \max_{\ell,m}\{R_{\ell,m}\}$.

5 Tensor isogeometric analysis

We use tensor notation to express the entries of Galerkin matrices. Then we consider kernel functions of the rank-R and derive a compact Kronecker format for the mass and stiffness matrices.

5.1 Mass and stiffness tensors

The (element-wise) integral

$$\mathbf{P} = \int_{\hat{\Omega}} \mathbf{T}(\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi} \quad \text{with elements} \quad p_{\boldsymbol{i}} = \int_{\hat{\Omega}} t_{\boldsymbol{i}}(\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi} \tag{22}$$

of a tensor-valued function $\mathbf{T} = \mathbf{T}(\boldsymbol{\xi}) = [t_{\boldsymbol{i}}(\boldsymbol{\xi})]$ with the variable $\boldsymbol{\xi} \in \hat{\Omega}$ is obtained by applying the integration to its entries. Integration commutes with multiplication in the following sense: the integral of a tensor product of d tensors $\mathbf{T}^{(k)}(\xi^{(k)})$, where the k-th one depends on $\xi^{(k)}$ only, inherits the product structure,

$$\int_{\hat{\Omega}} \bigotimes_{k=1}^{d} \mathbf{T}^{(k)}(\xi^{(k)})\,\mathrm{d}\boldsymbol{\xi} = \bigotimes_{k=1}^{d} \int_{a_k}^{b_k} \mathbf{T}^{(k)}(\xi^{(k)})\,\mathrm{d}\xi^{(k)}. \tag{23}$$

We define the *mass tensor*

$$\mathbf{M} = \int_{\hat{\Omega}} \omega \underbrace{\mathbf{B}}_{=[\beta_{\boldsymbol{i}}]} \otimes \underbrace{\mathbf{B}}_{=[\beta_{\boldsymbol{j}}]} \, \mathrm{d}\boldsymbol{\xi} \quad \in \mathbb{W}_{(\boldsymbol{n},\boldsymbol{n})}, \tag{24}$$

where the scalar $\omega = |\det \nabla G|$ is determined by the geometry mapping. Note that the integrand is a tensor, see (22). The associated *mass matrix* $M = \mathrm{mat}\,(\mathbf{M}) \in \mathbb{R}^{\pi(\boldsymbol{n})\times\pi(\boldsymbol{n})}$ is obtained by performing matricization with respect to the indices $\boldsymbol{i}$ and $\boldsymbol{j}$.

In addition we consider the *stiffness tensor*

$$\mathbf{S} = \int_{\hat{\Omega}} [K \cdot (\nabla \otimes \underbrace{\mathbf{B}}_{=[\beta_{\boldsymbol{i}}]})] \cdot (\nabla \otimes \underbrace{\mathbf{B}}_{=[\beta_{\boldsymbol{j}}]}) \, \mathrm{d}\boldsymbol{\xi} \quad \in \mathbb{W}_{(\boldsymbol{n},\boldsymbol{n})}, \tag{25}$$

where the matrix $K = [\kappa_{\ell,m}] \in \mathbb{R}^{d\times d}$ is again determined by the geometry mapping, see (7). The *stiffness matrix* $S = \mathrm{mat}\,\mathbf{S} \in \mathbb{R}^{\pi(\boldsymbol{n})\times\pi(\boldsymbol{n})}$ is the matricization of the stiffness tensor with respect to the indices $\boldsymbol{i}$ and $\boldsymbol{j}$.

We rewrite the stiffness tensor in the form

$$\mathbf{S} = \sum_{\ell,m=1}^{d} \mathbf{P}_{\ell,m} \ , \tag{26}$$

where we use the auxiliary tensors

$$\mathbf{P}_{\ell,m} = \int_{\hat{\Omega}} \kappa_{\ell,m} \, (\frac{\partial}{\partial \xi^{(\ell)}} \mathbf{B}) \otimes (\frac{\partial}{\partial \xi^{(m)}} \mathbf{B}) \, \mathrm{d}\boldsymbol{\xi} \quad \in \mathbb{W}_{(\boldsymbol{n},\boldsymbol{n})} \quad \text{for} \quad \ell, m = 1, \ldots, d \ . \tag{27}$$

Using the representation (26) will be more convenient for splitting the integration into univariate ones.

For later reference we introduce an artificial representation of the partial derivative operator as a product of d factors,

$$\frac{\partial}{\partial \xi^{(\ell)}} = \prod_{k=1}^{d} \vartheta_{\ell}^{(k)} \quad \text{where} \quad \vartheta_{\ell}^{(k)} = \begin{cases} \dfrac{\partial}{\partial \xi^{(\ell)}} & \text{if} \quad k = \ell \\ \mathrm{id} & \text{otherwise,} \end{cases} \tag{28}$$

simply by multiplying it with several copies of the identity operator.

5.2 The low rank case

We now consider mass and stiffness tensors, which are defined by rank-R scalar- and matrix-valued functions, see (20) and (21). More precisely, the scalar factor ω and the matrix-valued function $K = [\kappa_{\ell,m}]_{\ell,m=1,\ldots,d}$ are now given as

$$\omega(\boldsymbol{\xi}) = \sum_{r=1}^{R} \prod_{k=1}^{d} \omega_r^{(k)}(\xi^{(k)}), \quad \kappa_{\ell,m}(\boldsymbol{\xi}) = \sum_{r=1}^{R} \prod_{k=1}^{d} \kappa_{\ell,m,r}^{(k)}(\xi^{(k)}). \tag{29}$$

We also collect the factors of the elements of the matrix K in factor matrices

$$K_r^{(k)}(\xi^{(k)}) = [\kappa_{\ell,m,r}^{(k)}(\xi^{(k)})]_{\ell,m=1,\dots,d}.$$

In Section 6 we will discuss how to obtain approximate low rank representations for K or ω, in the case that they are not given a priori in this way.

Using these representations makes it possible to reduce the integral evaluations to the univariate case. This will be expressed with the help of auxiliary matrices or tensors.

More precisely, we consider a univariate function $w : [a_k, b_k] \to \mathbb{R}$, a univariate matrix-valued function

$$W = [\omega_{\ell,m}]_{\ell,m=1,\dots,d} : [a_k, b_k] \to \mathbb{R}^{d\times d},$$

a B-spline basis (18), and the factorization (28). We now define the *univariate mass matrix* with weight w,

$$M^{(k)}\{w\} = \int_{a_k}^{b_k} w\, \boldsymbol{\beta}^{(k)} \otimes \boldsymbol{\beta}^{(k)}\, \mathrm{d}\xi^{(k)} \qquad \in \mathbb{R}^{n_k\times n_k} \tag{30}$$

and the *univariate auxiliary matrices* with weights $W = [\omega_{\ell,m}]_{\ell,m=1,\dots,d}$,

$$P_{\ell,m}^{(k)}\{W\} = \int_{a_k}^{b_k} \omega_{\ell,m}\, (\vartheta_\ell^{(k)}\boldsymbol{\beta}^{(k)}) \otimes (\vartheta_m^{(k)}\boldsymbol{\beta}^{(k)})\, \mathrm{d}\xi^{(k)} \qquad \in \mathbb{W}_{(n_k,n_k)}. \tag{31}$$

We are now in position to state our main result regarding the structure of Galerkin matrices in the low-rank case.

Theorem 1 (Kronecker format)**.** *The mass and stiffness matrices possess the Kronecker representation*

$$M = \sum_{r=1}^{R} \bigotimes_{k=1}^{d} M^{(k)}\{\omega_r^{(k)}\} \qquad \textit{and} \qquad S = \sum_{r=1}^{R} \sum_{\ell,m=1}^{d} \bigotimes_{k=1}^{d} P_{\ell,m}^{(k)}\{K_r^{(k)}\}\,, \tag{32}$$

provided that ω and K are rank-R functions as defined in (29).

Proof. For the mass tensor, starting from (24) and in view of (18) and (29) we have

$$\mathbf{M} = \sum_{r=1}^{R} \int_{\hat\Omega} \left(\prod_{k=1}^{d} \omega_r^{(k)}\right) \left(\bigotimes_{k=1}^{d} \boldsymbol{\beta}^{(k)}\right) \otimes \left(\bigotimes_{k=1}^{d} \boldsymbol{\beta}^{(k)}\right) \mathrm{d}\boldsymbol{\xi}\,.$$

By reordering the factors of the two d-fold tensor products we get

$$\mathbf{M} \cong \sum_{r=1}^{R} \int_{\hat\Omega} \left(\prod_{k=1}^{d} \omega_r^{(k)}\right) \left(\bigotimes_{k=1}^{d} \left[\boldsymbol{\beta}^{(k)} \otimes \boldsymbol{\beta}^{(k)}\right]\right) \mathrm{d}\boldsymbol{\xi}\,.$$

Now we use (12) and obtain

$$\mathbf{M} \cong \sum_{r=1}^{R} \int_{\hat\Omega} \bigotimes_{k=1}^{d} \left(\omega_r^{(k)}\, \boldsymbol{\beta}^{(k)} \otimes \boldsymbol{\beta}^{(k)}\right) \mathrm{d}\boldsymbol{\xi} = \sum_{r=1}^{R} \bigotimes_{k=1}^{d} \int_{a_k}^{b_k} \omega_r^{(k)}\, \boldsymbol{\beta}^{(k)} \otimes \boldsymbol{\beta}^{(k)}\, \mathrm{d}\xi^{(k)}\,,$$

where we swapped integration and tensor product, cf. (23). Next we use (30) to confirm

$$\mathbf{M} \cong \sum_{r=1}^{R} \bigotimes_{k=1}^{d} M^{(k)}\{\omega_r^{(k)}\}. \tag{33}$$

Now we consider the stiffness tensor. First we note that the factorization (28) and (12) implies

$$\frac{\partial}{\partial \xi^{(\ell)}} \mathbf{B} = \left(\prod_{k=1}^{d} \vartheta_\ell^{(k)} \right) \left(\bigotimes_{k=1}^{d} \boldsymbol{\beta}^{(k)} \right) = \prod_{k=1}^{d} \left(\vartheta_\ell^{(k)} \boldsymbol{\beta}^{(k)} \right).$$

Combining this result with the low rank representation (29) we expand the slices of the auxiliary tensors (27) as

$$\mathbf{P}_{\ell,m} \cong \sum_{r=1}^{R} \bigotimes_{k=1}^{d} \int_{a_k}^{b_k} \kappa_{\ell,m,r} \, (\vartheta_\ell^{(k)} \boldsymbol{\beta}^{(k)}) \otimes (\vartheta_m^{(k)} \boldsymbol{\beta}^{(k)}) \, \mathrm{d}\xi^{(k)} .$$

Next we use (31) and (26) to confirm

$$\mathbf{S} \cong \sum_{r=1}^{R} \sum_{\ell,m=1}^{d} \bigotimes_{k=1}^{d} P_{\ell,m}^{(k)}\{K_r^{(k)}\}. \tag{34}$$

Based on the representations (33) and (34) of the mass and stiffness tensors, we finally exploit the relationship between Kronecker and tensor product (16). This confirms the representations (32) of the mass and stiffness matrices, which are specific matricizations of the associated tensors. □

A completely analogous formulation can be derived for the advection matrix (8), or for other kinds of matrices appearing in a Galerkin discretization.

We will refer to the representation (32), which expresses the mass and stiffness matrices as a sum of Kronecker products of smaller matrices, which are obtained by univariate integrations, as the *Kronecker format* of the matrices. Moreover, the number of summands will be referred to as the *Kronecker rank* of the matrix, see also [**?**]. Note that this notion of rank is different from the usual matrix rank.

Example 2. Consider a volumetric domain Ω, given as the image of the parametric map

$$G : [0,1]^3 \to \mathbb{R}^3 \quad \text{with} \quad (x,y,z) \quad \mapsto \quad G(\boldsymbol{\xi}) = \Big((x^2-1)(y+1), (x+1)(y+1), z \Big).$$

where we use the abbreviations $\xi^{(1)} = x$, $\xi^{(2)} = y$, $\xi^{(3)} = z$. For the sake of simplicity we work with the monomial basis, instead of using B-splines. We consider a single element $[0,1]^3$ and univariate quadratic basis $(\beta_0, \beta_1, \beta_2) = (1, t, t^2)$, $t \in [0,1]$ in all directions. The tensor-product basis functions are

$$\mathbf{B} = [\beta_{\boldsymbol{i}}] = [\beta_{i_1} \beta_{i_2} \beta_{i_3}] = \left[x^{i_1} y^{i_2} z^{i_3} \right] \quad , \quad \boldsymbol{i} = (i_1, i_2, i_3) \in \{0,1,2\}^3 .$$

We have $\omega(\boldsymbol{\xi}) = \det \nabla G = (x+1)^2(y+1)$, which is clearly a rank-1 function. Therefore the entries of the mass tensor are integrals of the form

$$m_{\boldsymbol{i},\boldsymbol{j}} = \int_{[0,1]^3} \omega(\boldsymbol{\xi})\ \beta_{\boldsymbol{i}}\,\beta_{\boldsymbol{j}}\,\mathrm{d}\boldsymbol{\xi} = \int_0^1 (x+1)^2 x^{i_1+j_1}\mathrm{d}x \int_0^1 (y+1) y^{i_2+j_2}\mathrm{d}y \int_0^1 z^{i_3+j_3}\mathrm{d}z\,.$$

Each univariate integral is easily computed either by quadrature or in closed form. By considering the vector of degrees of freedom $\operatorname{vec}\mathbf{B} = (1, z, z^2, \ldots, x^2y^2z^2)^\top$, we obtain the Kronecker format (32) of the mass matrix $M \in \mathbb{R}^{27\times 27}$,

$$M = M_1 \otimes M_2 \otimes M_3 = \begin{bmatrix} 7/3 & 17/12 & 31/30 \\ 17/12 & 31/30 & 49/60 \\ 31/30 & 49/60 & 71/105 \end{bmatrix} \otimes \begin{bmatrix} 3/2 & 5/6 & 7/12 \\ 5/6 & 7/12 & 9/20 \\ 7/12 & 9/20 & 1/30 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix},$$

where the Kronecker factors are indexed by quadratic monomial bases. In the same spirit, we can see that the rank of each entry of $K(\boldsymbol{\xi})$ is at most 2,

$$K = \frac{1}{\omega}\begin{bmatrix} (x+1)^2 + (y+1)^2 & (1-x)(1+x)^2 - 2x(y+1)^2 & 0 \\ (1-x)(1+x)^2 - 2x(y+1)^2 & (x^2+1)^2 + 4x^2y(y+2) & 0 \\ 0 & 0 & \omega^2 \end{bmatrix},$$

that is, the stiffness matrix has Kronecker rank equal to 9. Therefore it can be represented by 27 matrices [1] of size 3×3. Note that this is only one third of the storage required by the full matrix, even in this toy example.

6 Efficient matrix assembly

We present the algorithm of matrix generation based on low rank approximation of integral kernels. It is assumed that the domain is given by the geometry map (1), where the basis functions are tensor-product B-splines of degree $\boldsymbol{p}$, which are defined on the knot vectors $\boldsymbol{\tau}_k$, $k = 1, \ldots, d$.

6.1 The algorithm

We consider the formation of the mass and the stiffness matrices. The approach can be adapted easily to other relevant matrices. The algorithm consists of three steps:

Step 1: Spline projection. We define two spline projectors Π' and Π'' which generate accurate representation of the kernels ω and K, respectively. Firstly we specify knot vectors $\boldsymbol{\gamma}_k$ and degrees $\boldsymbol{q} = (q_1, \ldots, q_d)$ for the target space of the projections. Secondly we define the projection of a given function by performing tensor-product interpolation

[1] Due to symmetry, 6 of these matrices appear twice, therefore storage reduces to 21 matrices.

at the Greville points. These procedures define rank-1 operators and can be performed efficiently, by solving a sequence of univariate interpolation problems.

In the case of the mass matrix, the kernel function $\omega = |\det \nabla G|$ is a tensor-product spline function in a spline space of higher degree and lower smoothness than the space associated to G. More precisely, the partial derivative $\frac{\partial}{\partial_k} G(\boldsymbol{\xi})$ is a spline function with differentiability reduced by one with respect to the knots in direction k. Overall, the determinant has degree $\boldsymbol{q} = d\boldsymbol{p} - \mathbf{1}$. To construct the knot vector $\boldsymbol{\gamma}_k$, we start from $\boldsymbol{\tau}_k$ and increase the multiplicity of each interior knot by one. Then we apply degree elevation (which further increases knot multiplicities) until the degree reaches $\boldsymbol{q}$. The projection Π' is simply interpolation with respect to the Greville points of this tensor-product spline space. We obtain

$$(\Pi'\omega)(\boldsymbol{\xi}) = \mathbf{V} : \mathbf{B}'(\boldsymbol{\xi}) = \omega(\boldsymbol{\xi}) \ . \tag{35}$$

For the case of the stiffness matrix, the numerator in (7) has degree $2(d-1)\boldsymbol{p}$. The continuity at the knots is reduced by one, due to the involved partial derivatives of G, as before. Moreover, the functions are now rational, therefore an approximation error occurs [2] . We observed that the approximation error reaches machine precision after few degree elevation steps, starting from the knot vectors $\boldsymbol{\gamma}_k$ used to define $\mathbf{B}'$. We obtain

$$(\Pi''\kappa_{\ell,m})(\boldsymbol{\xi}) = \mathbf{C}_{\ell,m} : \mathbf{B}''(\boldsymbol{\xi}) \approx \kappa_{\ell,m}(\boldsymbol{\xi}) \ , \quad \ell, m = 1, \dots, d \ . \tag{36}$$

Step 2: Low rank approximation. We use the spline representations created by Π' and Π'' to construct low rank approximations of ω and $\kappa_{\ell,m}$. In particular, we consider canonical tensor decompositions of the coefficient tensors, (35) and (36)

$$\mathbf{V} \approx T'_{R'}\mathbf{V} = \sum_{r=1}^{R'} \bigotimes_{k=1}^{d} \mathbf{v}_r^{(k)} \quad \text{and} \quad \mathbf{C}_{\ell,m} \approx T''_{R''}\mathbf{C}_{\ell,m} = \sum_{r=1}^{R''} \bigotimes_{k=1}^{d} \mathbf{c}_{\ell,m,r}^{(k)} \tag{37}$$

which are obtained by applying the tensor truncation operators $T'_{R'}$ and $T''_{R''}$, see (17). The choice of the *approximate Kronecker ranks* R' and R'' will be addressed in Section 6.3. Consequently, we obtain low rank approximations of ω and K

$$\omega(\boldsymbol{\xi}) \approx (T'_{R'}\Pi'\omega)(\boldsymbol{\xi}) = (T'_{R'}\mathbf{V}) : \mathbf{B}'(\boldsymbol{\xi}) = \sum_{r=1}^{R'} \prod_{k=1}^{d} \omega_r^{(k)}(\xi^{(k)}) \tag{38}$$

and

$$\kappa_{\ell,m}(\boldsymbol{\xi}) \approx (T''_{R'}\Pi''\kappa_{\ell,m})(\boldsymbol{\xi}) = (T''_{R'}\mathbf{C}_{\ell,m}) : \mathbf{B}''(\boldsymbol{\xi}) = \sum_{r=1}^{R''} \prod_{k=1}^{d} \kappa_{\ell,m,r}^{(k)}(\xi^{(k)}) \tag{39}$$

where

$$\omega_r^{(k)}(\hat{x}^{(k)}) = \mathbf{v}_r^{(k)} \cdot \boldsymbol{\beta}'^{(k)}(\xi^{(k)}) \quad \text{and} \quad \kappa_{\ell,m,r}^{(k)}(\xi^{(k)}) = \mathbf{c}_{\ell,m,r}^{(k)} \cdot \boldsymbol{\beta}''^{(k)}(\xi^{(k)}) \ .$$

[2] One can eliminate this error by considering a NURBS with weight function ω. This, however, makes it more complicated to perform the low rank approximation in Step 2, except if the weight function has rank 1. Nevertheless this would not eliminate the low rank approximation error.

Step 3: Matrix Assembly. With the representations (38) and (39) at hand, we use Theorem 1 to evaluate the matrices. The assembly requires two ingredients. Firstly, one needs a procedure that can generate each univariate B-spline mass, stiffness and advection matrices, which appear as building blocks in the Kronecker format (32). Secondly, it requires a procedure that evaluates the sum of Kronecker products of matrices.

6.2 Computational complexity

In order to simplify the presentation we consider a $d-$dimensional tensor-product basis with $\boldsymbol{n} = (n_1, \ldots, n_d)$, $n_k \approx n$, and polynomial degree $p_k \approx p$, $k = 1, \ldots, d$. Moreover we assume that the projection operators used in the first step of Paragraph 6.1 are based on spline bases with $O(m^d)$ functions, for some positive integer m.

The number of non-zero entries, which equals $O(n^d p^d)$, is a lower bound for the time complexity of matrix assembly. When using a typical element-wise assembly procedure, one has to iterate over the $O(n^d)$ elements, and to compute $O(q^d p^{2d})$ values and derivatives of basis functions (where q stands for the number of quadrature points per direction) per element. This results in total costs of $O(n^d p^{3d})$ for the matrix formation, since $q = O(p)$.

We now discuss the complexity of matrix assembly based on low-rank tensor approximation, using the notions of Kronecker format and Kronecker rank, see Section 5.2.

Theorem 3. *Consider a B-spline discretization with $O(n^d)$ degrees of freedom on a patch with an approximate Kronecker rank R of the mass and stiffness matrices. The computation of the Kronecker format of these matrices requires $O(Rdnp^3 + Rdm^d)$ elementary operations, while the cost of converting it to the full matrix format amounts to $O(Rdn^d p^d)$.*

Proof. The spline projectors are based on interpolation, where we exploit the tensor-product structure. The cost of the interpolation problem is reduced to the cost of computing the LU factorization of d banded matrices of size $m \times m$ with bandwidth $O(p)$ each, which is done in $O(dmp^2)$.

The tensor decomposition is applied to the integral kernels, which are discretized (by the spline projectors) on a tensor grid of dimension $\boldsymbol{m} = (m, \ldots, , m)$, i.e. we evaluate the functions at $O(m^d)$ points. The cost of this decomposition is dominated by the HOSVD algorithm, i.e. by the cost of performing matrix SVD on the d unfoldings of a tensor of size $O(m^d)$, with a total cost of $O(dm^{d+1})$, cf. [**?**,**?**]. This cost can be further reduced to $O(Rdm^d)$ if we employ reduced SVD computation [3], i.e. if we compute only the R dominant singular values, cf. [**?**,**?**].

We generate the each univariate matrix in (32) of Theorem 1 using $O(np^3)$ elementary operations. Since the matrix is a sum of R Kronecker products of such matrices, the total cost for computing the Kronecker format is $O(Rdnp^3)$, where we use $O(p)$ quadrature points in each knot-span. Finally, for computing the $n^d \times n^d$ matrix, it suffices to perform the Kronecker product computations in (32), with complexity $O(Rdn^d p^d)$. □

[3] Probabilistic algorithms reduce the cost of rank-R SVD even to $O(m^d \log R)$ [**?**].

The costs of the Kronecker product computation dominate the overall complexity. Indeed we have $m = O(n)$. Moreover we obtain $m \ll n$ if the geometry representation uses a coarser spline space than the isogeometric discretization. This can be achieved by employing a larger polynomial degree for the spline projection, due to the built-in smoothness of the geometry.

One important feature of this decomposition is that relatively simple domains will produce a low rank representation, while the rank will be larger for more complicated ones. Thus the complexity of the computation follows the intrinsic geometric complexity of the domain and of its parameterization. Moreover, if the bilinear form is symmetric (e.g. a Laplacian) the algorithm will produce a symmetric matrix. In the case of the stiffness matrix the symmetry can be used to further reduce the computation and storage costs, since certain matrix factors appear twice in the Kronecker format.

The method offers an additional advantage in the case when the Kronecker format of the matrix suffices for the further computations. This advantage is twofold; both storage cost and computation times reduce significantly. With respect to storage, the Kronecker format requires storing $O(Rdnp)$ values instead of $O(Rdn^d p^d)$ for the expanded matrix. As an instance of computation times we mention the matrix-vector product. The cost of this operation using the Kronecker format is $O(Rdpn^d)$ instead of $O(p^d n^d)$ for the full matrix, which is $O(Rdp)$ per vector element instead of $O(p^d)$ (where the vector has size $O(n^d)$). Therefore, isogeometric iterative solvers relying on this operation, may benefit from using this representation, see also [**?**].

6.3 Controlling the consistency error

Recall that the notion of consistency error refers to the error introduced during the matrix generation process. For our method, it depends on the spline projection error (introduced by Π) and the truncation error (introduced by the low-rank tensor approximation T_R). The overall error introduced in the function f, and consequently in the entries of a matrix involving f as integral kernel, is bounded by

$$\|f - T_R \Pi f\|_\infty \leq \varepsilon_\Pi + \varepsilon_{T_R} .$$

With this error bound at hand, the same approach as in [**?**, Th. 13] can be employed to derive consistency error bounds for Galerkin discretization of the particular bilinear forms. We shall verify experimentally in Section 7 that optimal convergence rates are obtained.

The usual scenario for controlling the error is to set an *a priori* error tolerance and to ensure that both contributions to the total error respect it. As already mentioned, in the general case the rank is related to the desired accuracy by $R = O(|\log \varepsilon_{T_R}|^{d-1} \log^{d-1} m)$, where m is the same as in Theorem 3. In particular in 2D, using R-truncated SVD implies an error bounded by

$$\varepsilon_{T_R} \leq \sqrt{\sum_{r>R} \sigma_r^2} ,$$

where σ_r are the singular values ordered by decreasing magnitude.

Regarding the projection error, in general it should be chosen such that the overall accuracy is maintained. For instance if Π uses uniform splines of degree q with m knot spans per direction, we have $\varepsilon_\Pi = O(m^{-q-1})$. To match, e.g., the L^2 discretization error, which is for elliptic problems $O(n^{-p-1})$ (for uniform grids), it suffices to choose $m = n^{(p+1)/(q+1)}$. Consequently for $q \approx dp$ we obtain $m \approx n^{1/d}$ as the degree p increases.

Finally, small (negligible) integration error is introduced by the quadrature of the 1D (mass or stiffness) integrals. The degree of these integrals is larger than $2p+1$, which is the exactness of the Gauss rule with $p+1$ nodes. Nevertheless, $p+1$ nodes per direction suffice to reduce the error down to the order of the approximation error.

The two approximation steps (projection and tensor decomposition) does not lead to rank deficiency of the resulting stiffness matrix (similarly for the mass matrix) provided that the procedure preserves positive definiteness (positivity) of the matrix $K(\boldsymbol{\xi})$ (of the factor ω). In [**?**] it is shown that these matrix properties can be preserved even for Kronecker rank-1 approximations. According to our numerical experiment in Section 7.2, any approximation with a rank lower than optimal did not lead to such rank deficiency, independently of the discretization step size h.

7 Experiments and numerical results

In this section we apply our method to a set of model shapes. We have developed a C++ implementation of the method in the G+Smo library [4] [**?**].

In what follows we examine the performance of tensor decomposition on the model patches, then we test the convergence rates obtained by our algorithm for problems involving the mass and the stiffness matrices, as well as the effect of rank truncation.

Moreover we compare computation times for matrix generation and for a simple conjugate gradient solver when using the full matrix format and when using the (low rank) Kronecker matrix format.

7.1 Tensor decomposition

In this section we examine the Kronecker-rank profile of some model geometries. The approximated quantities are the kernels appearing in the mass (6) and stiffness (7) integrals. The patches tested are shown in Table 2. For each shape, the degree $\boldsymbol{p}$ of the parameterization and the size of the control grid is indicated. For the functions ω, K, the obtained rank values and the size of the control grid of the representation used are given. This representation follows the spaces defined in Section 6.1.

We use HOSVD to obtain a Tucker representation of the tensors and we convert it to a canonical representation. For obtaining an even lower canonical rank, we use ALS iteration in the process. We use successive rank-1 approximations of the residue as initial values for each step of the ALS iteration. In all computations, we set the accuracy to $\varepsilon = 10^{-8}$, and we stop the iteration as soon as the Frobenius norm of the residue is below ε.

[4] *Geometry plus simulation modules*, http://gs.jku.at/gismo, 2016.

Table 2 shows that the rank profile is related to the geometric complexity of the domain. We know that in 2D the rank (computed by SVD) is limited by the minimum number of basis functions in the coordinate bases of the projection basis used in Π. In 3D, the rank is bounded by the product of the two short dimensions in the core tensor of the HOSVD. Below the shape we note the (tensor) dimensions of the B-spline basis used for the geometry parameterization, as well as for projecting Ω and K, respectively.

The results reveal that the overall rank profile is correlated with the geometric complexity of the model, rather than the polynomial degree. In particular, patches which are more symmetric have lower rank numbers, while irregular shapes (such as case (e)) have more elevated rank numbers, even though the degree of the parameterization is small.

We can observe that symmetric shapes or shapes with a polar parameterization have smaller rank values than shapes with arbitrary form. The correlation to the degree of the parameterization is weaker. Indeed, the trilinear shape in (2e) has higher rank values than (2b), which is quadratic in all parametric directions. Note however that the bottom and top faces of (2e) are arranged in an arbitrary fashion, contrarily to the mirror symmetry which is present in (2b).

We also tested the behaviour of the rank values when applying $h-$refinement to the projection space. In all cases it is verified that the computed rank value is invariant under $h-$refinement. This is an indication that these values are intrinsic properties of the underlying continuous functions ω or K, which in turn depend on the given parameterization.

7.2 Order of convergence

In this section we use the models (2c) and (2f) and we consider the problem of approximating the function

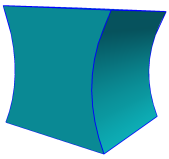
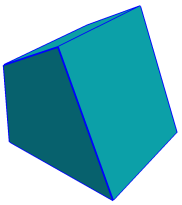
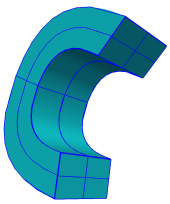
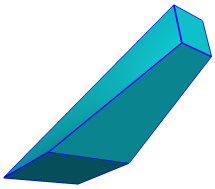
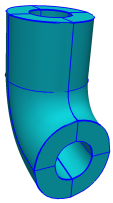
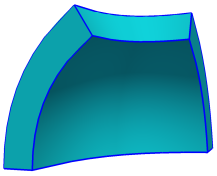
$$f(x, y, z) = \sin(\pi x) \sin(\pi y) \sin(\pi z) \ , \tag{40}$$

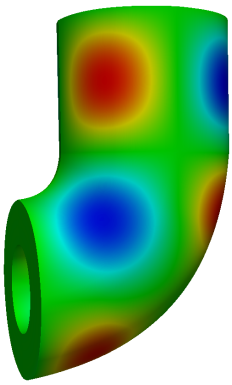
in two scenarios. In the first one, we use the mass matrix for computing the L^2 projection of $f(x, y, z)$ to the ($k-$refined) B-spline space obtained from the geometry representation. In the second scenario we consider the PDE $-\Delta f = g$, $f = f_0$ on $\partial\Omega$, where f_0 and g are defined so that the exact solution is (40). This PDE is solved for the control points of the approximate solution using the computed stiffness matrix. A view of the function on the two test domains is depicted in Figure 2.

As outlined in Section 6.3, the overall error is the sum of the discretization plus the consistency (integration) error. We verify that the overall error is of the order of the discretization error in Figure 3. Indeed, for both scenarios optimal orders of approximation are obtained, in the L^2 norm and the H^1 seminorm, respectively.

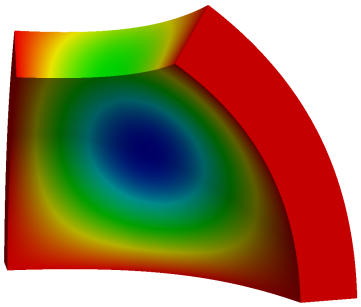
In a second experiment, we investigate the influence of rank truncation in the convergence rates. In particular, we obtain different approximations of $\omega = \det \nabla G$ using canonical rank $R = 1, 2, 3, 4$. The last value is equal to ω up to machine precision. In Figure 4 we present a convergence test of the different rank values. A plateau is

Table 2
Different patches and their rank profile. Shape: the degree and the number of coefficients per direction is reported. $\omega(\boldsymbol{\xi})$: The rank of the function and the size of the spline coefficient tensor used for representing the function. $K(\boldsymbol{\xi})$: The rank of each entry of the matrix, and the size of the spline coefficient tensor used for representing it. We remark that rank 0 is assigned to the null function.

Shape	Rank profile $\omega(\boldsymbol{\xi})$	Rank profile $K(\boldsymbol{\xi})$	Shape	Rank profile $\omega(\boldsymbol{\xi})$	Rank profile $K(\boldsymbol{\xi})$
(a)  $\boldsymbol{p} = (1,1,2)$, $2 \times 2 \times 3$	1 $6 \times 6 \times 3$	$\begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$ $5 \times 5 \times 9$	(d)  $\boldsymbol{p} = (1,1,1)$, $2 \times 2 \times 2$	2 $3 \times 3 \times 3$	$\begin{bmatrix} 4 & 4 & 0 \\ 4 & 4 & 0 \\ 0 & 0 & 2 \end{bmatrix}$ $5 \times 5 \times 5$
(b)  $\boldsymbol{p} = (2,2,2)$, $4 \times 4 \times 4$	1 $11 \times 11 \times 11$	$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $17 \times 17 \times 17$	(e)  $\boldsymbol{p} = (1,1,1)$, $2 \times 2 \times 2$	2 $3 \times 3 \times 3$	$\begin{bmatrix} 3 & 3 & 3 \\ 3 & 8 & 4 \\ 3 & 4 & 4 \end{bmatrix}$ $5 \times 5 \times 5$
(c)  $\boldsymbol{p} = (2,1,2)$, $9 \times 2 \times 5$	4 $24 \times 3 \times 12$	$\begin{bmatrix} 6 & 6 & 1 \\ 6 & 6 & 1 \\ 1 & 1 & 9 \end{bmatrix}$ $36 \times 5 \times 18$	(f)  $\boldsymbol{p} = (2,2,1)$, $3 \times 3 \times 2$	1 $6 \times 6 \times 3$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 2 \end{bmatrix}$ $9 \times 9 \times 5$



(c)



(f)

Fig. 2. The benchmark domains colored by the value of the function (40).

observed, which indicates that the approximation error is limited by the error in the rank approximation. A higher rank value sets this limit to higher accuracy. For the value $R = 4$, the full convergence rate is obtained. This shows that the rank truncation error is carried over to the final solution, therefore it has to be controlled with care to obtain high accuracy. We used a conjugate gradient method to solve the resulting linear

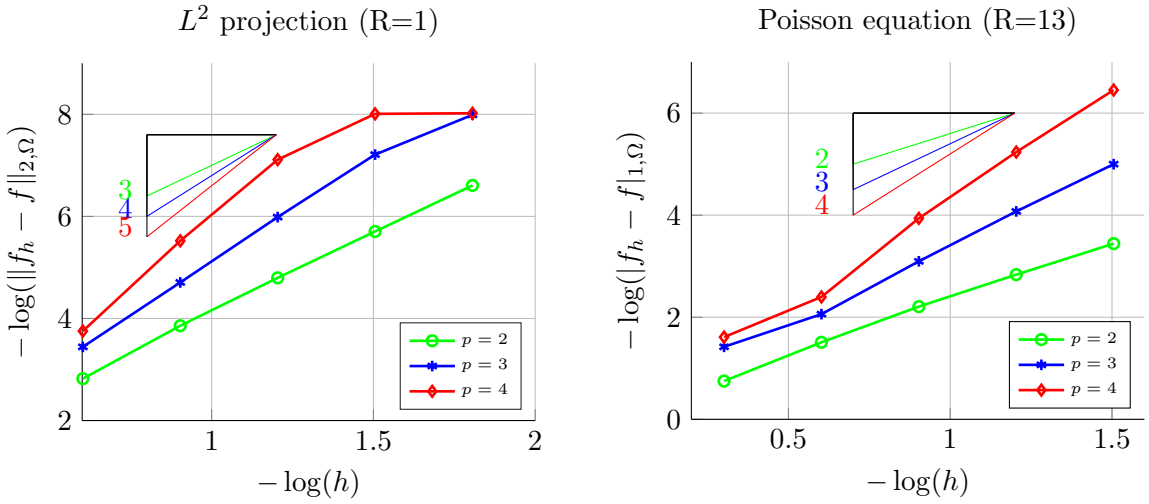


Fig. 3. Experimental orders of convergence for the L^2 projection and Laplacian problems using low rank mass and stiffness approximation, for degrees $p = 2, 3, 4$. The domain Ω is the volume depicted in Table 2f, and f_h stands for the computed approximation of (40).

systems, and no rank deficiency phenomena were observed in the process.

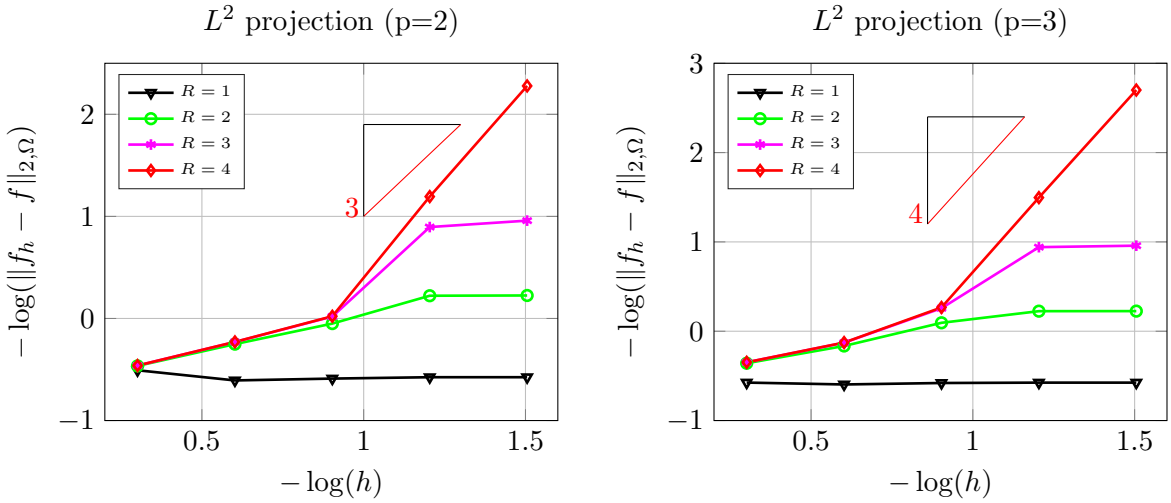


Fig. 4. Rank truncation effect: We apply L^2 projection with approximate mass matrix using different ranks, for degrees $p = 2$ and $p = 3$. The domain used is the one shown in Table 2c.

7.3 Computation and storage costs for matrix assembly

In this section we examine the computation and storage costs of the two different representations. First, we look at the Kronecker matrix format (KF) and secondly we use our algorithm to compute the (usual) standard format (SF) of the matrix, by taking Kronecker products (KP).

Moreover, we compare our assembly method with a classical tensor-product Gauss quadrature (TPG) approach with $p + 1$ nodes per parametric direction, where p is the

B-spline degree in each direction. The experimentally observed timings and speedup are shown in Table 2.

We observe that overall computation time is negligible for the KF format, when compared to the time to compute the KP and SF. The timings include the tensor decomposition as well as the generation of the univariate B-spline mass or stiffness matrices in KF. This is a striking result; assembly times can be logarithmic with respect to the number of degrees, if solely the KF suffices for further use. Moreover, the storage costs for KF grow linearly under global $h-$refinement. For instance, for the last step of refinement in (2c), we would need around 45GB of RAM for storing the matrix at the last step of refinement. This was beyond available memory, therefore only KF is computed.

Regarding comparison with respect to TPG, in all experiments a significant speedup is obtained for both (the SF of) the mass and the stiffness matrix computation. The speedup is asymptotically constant with respect to n, as expected (computation time grows linearly with respect to n^d). Moreover, speedup doubles each time we increase the polynomial degree by one. Finally, the speedup certainly decreases when the rank R increases; however, even in the case of $R = 37$ the new method is 10 times faster than the classical algorithm. Already from these small (quadratic and cubic) degrees a great advantage becomes visible. Certainly, speedups become even more prominent for higher polynomial degree. Nevertheless, the speedup obtained for low degree splines is most important, due to the common use of these degrees in practical applications.

Table 3
For the patch 2c. The Kronecker rank of the mass matrix is 4

DoFs	Storage (NNZ)		Computation (sec)			Speedup	
(mass, $p = 2$)	KF	SF	KF	KP	TPG	TPG/(KF+KP)	TPG/KF
$37 \times 10 \times 19$	1,217	662,244	0.005	0.032	0.427	11.5	79.8
$69 \times 18 \times 35$	2,337	4,671,324	0.006	0.217	3.430	15.4	616.9
$133 \times 34 \times 67$	4,577	35,019,084	0.006	1.597	24.765	15.5	4,259
$261 \times 66 \times 131$	9,057	271,049,004	0.008	11.885	190.862	16.1	24,681
$517 \times 130 \times 259$	18,017	2,132,574,444	0.008	-	-	-	-
(mass, $p = 3$)							
$38 \times 11 \times 20$	1,757	2,031,120	0.005	0.090	1.947	20.4	357.8
$70 \times 19 \times 36$	3,325	13,592,656	0.006	0.582	14.814	25.2	2,605
$134 \times 35 \times 68$	6,461	99,034,320	0.007	4.241	110.754	26.1	16,885
$262 \times 67 \times 132$	12,733	755,219,920	0.007	30.033	881.372	29.3	125,894
$518 \times 131 \times 260$	25,277	5,897,023,440	0.009	-	-	-	-

7.4 Comparison of iterative solving in Kronecker and full format

One advantage of the KF representation is its minimal storage requirements. This can be advantageous when available resources are forbidding for computing the SF of the matrix. In this section we demonstrate the possibility to solve the linear system directly using the KF as a black-box matrix, and refrain from computing the expanded matrix. To do so, we perform matrix-vector product where the matrix is in KF.

Table 4
For the patch 2f. The mass matrix has Kronecker rank 1.

DoFs	Storage (NNZ)		Computation (sec)			Speedup	
(mass, $p = 2$)	KF	SF	KF	KP	TPG	TPG/(KF+KP)	TPG/KF
$10 \times 10 \times 10$	133	85,184	0.001	0.003	0.054	15.0	56.3
$18 \times 18 \times 18$	253	592,704	0.001	0.016	0.415	23.9	439.7
$34 \times 34 \times 34$	493	4,410,944	0.002	0.122	3.35	27.2	2,024
$66 \times 66 \times 66$	973	34,012,224	0.002	0.948	24.6	25.9	12,527
$130 \times 130 \times 130$	1,933	267,089,984	0.002	7.363	188.8	25.6	122,133
(mass, $p = 3$)							
$11 \times 11 \times 11$	196	274,625	0.001	0.009	0.24	24.8	247.8
$19 \times 19 \times 19$	364	1,771,561	0.001	0.050	1.94	38.3	1,895
$35 \times 35 \times 35$	700	12,649,337	0.001	0.336	14.7	43.6	13,221
$67 \times 67 \times 67$	1,372	95,443,993	0.001	2.511	109.4	43.5	84,872
$131 \times 131 \times 131$	2,716	741,217,625	0.002	18.066	873.4	48.3	509,198

Table 5
For the patch 2c. The stiffness matrix has Kronecker rank 37.

DoFs	Storage (NNZ)		Computation (sec)			Speedup	
(stiffness, $p = 2$)	KF	SF	KF	KP	TPG	TPG/(KF+KP)	TPG/KF
$37 \times 10 \times 19$	11,249	662,244	0.050	0.17	0.691	3.15	13.9
$69 \times 18 \times 35$	21,609	4,671,324	0.052	1.214	5.522	4.36	107
$133 \times 34 \times 67$	42,329	35,019,084	0.056	9.394	38.34	4.06	689.5
$261 \times 66 \times 131$	83,769	271,049,004	0.061	66.3	311.9	4.7	5,134
$518 \times 131 \times 260$	233,804	5,897,023,440	0.068	-	-	-	-
(stiffness, $p = 3$)							
$38 \times 11 \times 20$	16,244	2,031,120	0.052	0.49	4.12	7.5	78.8
$70 \times 19 \times 36$	30,748	13,592,656	0.053	3.34	29.2	8.6	552.5
$134 \times 35 \times 68$	59,756	99,034,320	0.057	23.1	229.6	9.9	3,997
$262 \times 67 \times 132$	117,772	755,219,920	0.066	172.7	1,852	10.7	27,965
$518 \times 131 \times 260$	233,804	5,897,023,440	0.084	-	-	-	-

We consider the L^2 projection problem of Paragraph 7.2 (using the rank-4 KF from Figure 2c). The moment vector and the rank-4 Kronecker product sum representation (exact up to machine precision) of the mass matrix are considered already computed. We examine two scenarios, for polynomial degrees $p = 2$ and $p = 3$. First, the Kronecker format is converted to the full matrix (i.e. we perform the Kronecker product computations) and then a conjugate gradient (CG) iteration is used to solve the system. Second, we use fast matrix-vector product between the Kronecker format of the mass matrix and the right-hand side vector. In both scenarios, we use a diagonal (Jacobi) preconditionner.

In Table 7 we report the computation times in these two scenarios. Also, in Table 8 we report the computation times in these two scenarios for computing the solution of the Poisson equation (using the rank-13 KF from Figure 2f).

In all cases, the CG residual is set below $\varepsilon = 10^{-10}$. This explains the higher iteration numbers for very coarse discretizations: in these cases the expected accuracy is much

Table 6
For the patch depicted in Table 2f. The stiffness matrix has Kronecker rank 13.

DoFs	Storage (NNZ)		Computation (sec)			Speedup	
(stiffness, $p = 2$)	KF	SF	KF	KP	TPG	TPG/(KF+KP)	TPG/KF
$10 \times 10 \times 10$	1,585	85,184	0.002	0.008	0.087	8.4	47.1
$18 \times 18 \times 18$	3,025	592,704	0.002	0.055	0.7	12.1	315.8
$34 \times 34 \times 34$	5,905	4,410,944	0.002	0.4	5.45	13.6	2,235
$66 \times 66 \times 66$	11,665	34,012,224	0.003	3.1	39.2	12.6	12,544
$130 \times 130 \times 130$	23,185	267,089,984	0.005	21.6	306.3	14.2	65,461
(stiffness, $p = 3$)							
$11 \times 11 \times 11$	2,341	274,625	0.002	0.025	0.5	18.7	250.9
$19 \times 19 \times 19$	4,357	1,771,561	0.002	0.153	4.08	26.2	1,812
$35 \times 35 \times 35$	8,389	12,649,337	0.003	1.086	29.4	27.0	10,596
$67 \times 67 \times 67$	16,453	95,443,993	0.004	8.142	226.5	27.8	58,659
$131 \times 131 \times 131$	32,581	741,217,625	0.006	58.707	1,802	30.7	289,588

lower that ε, therefore the many iterations approximate remaining noise down to ε. In any case, our focus here is on the space requirements and the computation time of the KF, instead of testing the performance of the solver. We refer the reader to [**?**,**?**] for a thorough presentation of solving linear systems and PDEs with tensor product structure.

The last row would required 2GB or 5.5GB of memory to execute the Kronecker product. Since these memory sizes are close to the limits of the system memory (6GB), we did not attempt to compute it. However, the Kronecker format performed perfectly fine, and CG converged in one minute, for degree 3.

Regarding computing times, the KF-CG outperforms SF-CG when the rank is sufficiently small. Even for larger rank values, increasing the degree is in favor of KF-CG. This can be explained, since higher degrees burden the sparsity of SF. Even for the degree value $p = 3$ and rank $R = 13$ as is the case in Table 8, when the number of degrees of freedom grow, KF-CG has comparable performance. It should be noted that our experiments we assumed no structure of the right-hand side. Therefore the complexity of the matrix-vector multiplication remains for both methods linear in the size of this vector, which is $O(n^d)$.

Table 7
Comparison between the classical matrix representation (SF) and the Kronecker product sum representation (KF) for the L^2 projection problem.

Degree $p = 2$				Degree $p = 3$			
DoFs	KF-CG	KP + SF-CG	ITER	DoFs	KF-CG	KP + SF-CG	ITER
$37 \times 10 \times 19$	.15	.026 + .245	155	$38 \times 11 \times 20$	.70	.076 + 2.31	505
$69 \times 18 \times 35$	.82	.192 + 1.33	126	$70 \times 19 \times 36$	3.19	.545 + 10.7	357
$133 \times 34 \times 67$	5.88	1.47 + 7.48	93	$134 \times 35 \times 68$	14.71	4.22 + 49.3	219
$261 \times 66 \times 131$	32.9	-	65	$262 \times 67 \times 132$	68.9	-	121

Table 8
Comparison between the classical matrix representation (SF) and the Kronecker product sum representation (KF) for the solution of the isogeometric Laplacian problem.

Degree $p = 2$				Degree $p = 3$			
DoFs	KF-CG	KP + SF-CG	ITER	DoFs	KF-CG	KP + SF-CG	ITER
$10 \times 10 \times 10$	.0208	.0084 + .0048	59	$11 \times 11 \times 11$	.078	.022+.035	137
$18 \times 18 \times 18$	.1854	.0557 + .0714	78	$19 \times 19 \times 19$	.494	.141+.396	144
$34 \times 34 \times 34$	1.920	.3766 + .9012	108	$35 \times 35 \times 35$	4.12	1.01+2.95	126
$66 \times 66 \times 66$	33.94	3.198 + 13.90	201	$67 \times 67 \times 67$	40.35	9.51+32.32	168

8 Conclusions

The global geometry map has been regarded many times in the literature as a drawback of the isogeometric paradigm, in terms of matrix assembly costs in 3D. Tensor decomposition methods transform this "curse of dimensionality" into a "blessing" for the efficient implementation of isogeometric analysis. Indeed, both memory requirements and computation times are drastically reduced, compared to the typical local approach inspired by finite element methods.

Several further improvements can be considered. For example, in our experiments we used a simple ALS iteration combined with a greedy strategy for canonical tensor decomposition. The rank values obtained by this algorithm are known to be suboptimal. The reason is that the optimization procedure may converge to local minima, due to the choice of initial values. More sophisticated methods exist, for instance one can enhance the algorithm with line search approaches to escape from local minima. Nevertheless the obtained sub-optimal rank values provided high efficiency gains in the assembly procedure.

An interesting possibility is to use the Kronecker format as the basic matrix format. This has a two-fold advantage, since not only it reduces drastically the matrix generation timings, but also the memory requirements reduce exponentially. In practice, this memory reduction allows to represent matrices will billions of rows without the need of several terabytes of memory. For example, this can be used for keeping the communication costs low in a distributed memory parallel environment. We also refer to the isogeometric tearing and interconnecting (IETI) solver, cf. [**?**] and references therein, where the performance of the parallelized method was limited by the available memory for storing the matrices, even though computation times remained very low. However, efficient linear solvers in this format is a topic fostering further investigations, which are beyond the aims of the present paper.

Finally, we note that typical volume parameterization such as Gordon-Coon's volumes or swept volumes [**?**] are expected to have low rank values and are favorable for our algorithm.

Acknowledgements

This research was supported by the National Research Network "Geometry + Simulation" (NFN S117, 2012–2016), funded by the Austrian Science Fund (FWF).

References

[1] M. Aigner, C. Heinrich, B. Jüttler, E. Pilgerstorfer, B. Simeon, and A. V. Vuong. *Mathematics of Surfaces XIII*, chapter Swept Volume Parameterization for Isogeometric Analysis, pages 19–44. Springer Berlin Heidelberg, 2009.

[2] P. Antolin, A. Buffa, F. Calabrò, M. Martinelli, and G. Sangalli. Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization. *Comp. Meth. Appl. Mech. Engrg.*, 285:817–828, 2015.

[3] F. Auricchio, L. Beirão da Veiga, T. J. R. Hughes, A. Reali, and G. Sangalli. Isogeometric collocation methods. *Mathematical Models and Methods in Applied Sciences*, 20(11):2075–2107, 2010.

[4] F. Auricchio, F. Calabrò, T. Hughes, A. Reali, and G. Sangalli. A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis. *Comp. Meth. Appl. Mech. Engrg.*, 249-252:15–27, 2012.

[5] M. Bartoň and V. M. Calo. Optimal quadrature rules for odd-degree spline spaces and their application to tensor-product-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 305:217 – 240, 2016.

[6] L. Beirão da Veiga, C. Lovadina, and A. Reali. Avoiding shear locking for the timoshenko beam problem via isogeometric collocation methods. *Computer Methods in Applied Mechanics and Engineering*, 241–244:38 – 51, 2012.

[7] P. Benner, R.-C. Li, and N. Truhar. On the ADI method for Sylvester equations. *Journal of Computational and Applied Mathematics*, 233(4):1035–1045, 2009.

[8] D. Braess. *Finite Elements: Theory, fast solvers and applications in solid mechanics, third edition*. Cambridge University Press, third edition, 2007.

[9] M. Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20 – 30, 2006. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

[10] S. Brenner and L. Scott. *The Mathematical Theory of Finite Element Methods*. Springer-Verlag, 2002.

[11] F. Buchegger, B. Jüttler, and A. Mantzaflaris. Adaptively refined multi-patch B-splines with enhanced smoothness. *Applied Mathematics and Computation*, 272, Part 1:159 – 172, 2016.

[12] F. Calabrò, G. Sangalli, and M. Tani. Fast formation of isogeometric Galerkin matrices by weighted quadrature. *arXiv preprint arXiv:1605.01238*, 2016.

[13] P. G. Ciarlet. *Finite Element Method for Elliptic Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[14] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, and V. Calo. The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers. *Computer Methods in Applied Mechanics and Engineering*, 213–216:353 – 361, 2012.

[15] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons, Chichester, England, 2009.

[16] C. de Boor. Efficient computer manipulation of tensor products. *ACM Trans. Math. Softw.*, 5(2):173–182, June 1979.

[17] L. de Lathauwer, B. de Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21:1253–1278, 2000.

[18] L. de Lathauwer, B. de Moor, and J. Vandewalle. On best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of high-order tensors. *SIAM J. Matrix Anal. Appl.*, 21:1324–1342, 2000.

[19] P. Dreesen, M. Ishteva, and J. Schoukens. Decoupling multivariate polynomials using first-order information and tensor decompositions. *SIAM Journal on Matrix Analysis and Applications*, 36(2):864–879, 2015.

[20] A. F., L. Beirão da Veiga, J. Kiendl, C. Lovadina, and A. Reali. Locking-free isogeometric collocation methods for spatial timoshenko rods. *Computer Methods in Applied Mechanics and Engineering*, 263:113 – 126, 2013.

[21] G. Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[22] L. Gao and V. M. Calo. Fast isogeometric solvers for explicit dynamics. *Comp. Methods Appl. Mech. Engrg.*, 274(0):19 – 41, 2014.

[23] L. Gao and V. M. Calo. Preconditioners based on the alternating-direction-implicit algorithm for the 2d steady-state diffusion equation with orthotropic heterogeneous coefficients. *Journal of Computational and Applied Mathematics*, 273(0):274 – 295, 2015.

[24] C. Giannelli, B. Jüttler, S. K. Kleiss, A. Mantzaflaris, B. Simeon, and J. Speh. THB-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 299:337 – 365, 2016.

[25] C. Giannelli, B. Jüttler, and H. Speleers. THB–splines: The truncated basis for hierarchical splines. *Computer Aided Geometric Design*, 29:485–498, 2012.

[26] H. Gomez, A. Reali, and G. Sangalli. Accurate, efficient, and (iso)geometrically flexible collocation methods for phase-field models. *Journal of Computational Physics*, 262(0):153 – 171, 2014.

[27] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[28] W. Hackbusch. *Tensor spaces and numerical tensor calculus*. Springer, Berlin, 2012.

[29] W. Hackbusch and B. Khoromskij. Low-rank Kronecker product approximation to multi-dimensional nonlocal operators. part I. Separable approximation of multi-variate functions. *Computing*, 76:177–202, 2006.

[30] W. Hackbusch, B. N. Khoromskij, and E. E. Tyrtyshnikov. Hierarchical Kronecker tensor-product approximations. *Journal of Numerical Mathematics*, 13(2):119–156, 2005.

[31] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, May 2011.

[32] R. R. Hiemstra, F. Calabrò, D. Schillinger, and T. J. Hughes. Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis. *ICES REPORT 16-11*, 2016.

[33] M. Hillman, J. Chen, and Y. Bazilevs. Variationally consistent domain integration for isogeometric analysis. *Comp. Meth. Appl. Mech. Engrg.*, 284:521–540, 2015.

[34] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *J. Math. Phys*, 6(1):164–189, 1927.

[35] C. Hofer. Parallelization of continuous and discontinuous Galerkin dual-primal isogeometric tearing and interconnecting methods. Technical Report No. 2016-01 https://www.dk-compmath.jku.at, DK Computational Mathematics Linz Report Series, 2016.

[36] T. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comp. Meth. Appl. Mech. Engrg.*, 194(39–41):4135–4195, 2005.

[37] T. Hughes, A. Reali, and G. Sangalli. Efficient quadrature for NURBS-based isogeometric analysis. *Comp. Meth. Appl. Mech. Engrg.*, 199(5 – 8):301–313, 2010.

[38] B. Jüttler, U. Langer, A. Mantzaflaris, S. Moore, and W. Zulehner. Geometry + simulation modules: Implementing isogeometric analysis. *Proc. Appl. Math. Mech.*, 14(1):961–962, 2014. Special Issue: 85th Annual Meeting of the Int. Assoc. of Appl. Math. and Mech. (GAMM), Erlangen 2014.

[39] V. Khoromskaia and B. N. Khoromskij. Tensor numerical methods in quantum chemistry: from Hartree-Fock to excitation energies. *Phys. Chem. Chem. Phys.*, 2015.

[40] B. N. Khoromskij. Structured rank-$(r_1, \ldots, r_d)$ decomposition of function-related operators in $\mathbb{R}^d$. *Comput. Meth. Appl. Math*, 6(2):194–220, 2006.

[41] B. N. Khoromskij. $\mathcal{O}(d \log n)$–Quantics approximation of N–d tensors in high-dimensional numerical modeling. *Constr. Appr.*, 34(2):257–280, 2011.

[42] B. N. Khoromskij. Tensor-structured numerical methods in scientific computing: survey on recent advances. *Chemometr. Intell. Lab. Syst.*, 110(1):1–19, 2012.

[43] B. N. Khoromskij and V. Khoromskaia. Low rank Tucker-type tensor approximation to classical potentials. *Central European journal of mathematics*, 5(3):523–550, 2007.

[44] B. N. Khoromskij and V. Khoromskaia. Multigrid accelerated tensor approximation of function related multidimensional arrays. *SIAM J. Sci. Comput.*, 31(4):3002–3026, 2009.

[45] T. Kolda and B. Bader. Tensor decompositions and applications. *SIAM Review*, 51/3:455–500, 2009.

[46] D. Kressner and C. Tobler. Krylov subspace methods for linear systems with tensor product structure. *SIAM Journal on Matrix Analysis and Applications*, 31(4):1688–1714, 2010.

[47] D. Kressner and A. Uschmajew. On low-rank approximability of solutions to high-dimensional operator equations and eigenvalue problems. *Linear Algebra and its Applications*, 493:556 – 572, 2016.

[48] U. Langer, A. Mantzaflaris, S. Moore, and I. Toulopoulos. Multipatch discontinuous Galerkin isogeometric analysis. In *Isogeometric Analysis and Applications*, Lecture Notes in Computational Science and Engineering, pages 1–32. Springer International Publishing, 2015.

[49] M. Los, M. Wozniak, M. Paszynski, L. Dalcin, and V. Calo. Dynamics with matrices possessing kronecker product structure. *Procedia Computer Science*, 51:286 – 295, 2015.

[50] A. Mantzaflaris and B. Jüttler. Integration by interpolation and look-up for Galerkin-based isogeometric analysis. *Comp. Methods Appl. Mech. Engrg.*, 284:373 – 400, 2015. Isogeometric Analysis Special Issue.

[51] A. Mantzaflaris, B. Jüttler, B. Khoromskij, and U. Langer. Matrix generation in isogeometric analysis by low rank tensor approximation. In J.-D. Boissonnat, A. Cohen, O. Gibaru, C. Gout, T. Lyche, M.-L. Mazure, and L. L. Schumaker, editors, *Curves and Surfaces*, volume 9213 of *LNCS*, pages 321–340. Springer, 2015.

[52] I. V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.

[53] G. Sangalli and M. Tani. Isogeometric preconditioners based on fast solvers for the Sylvester equation, 2016.

[54] D. Schillinger, L. Dedè, M. Scott, J. Evans, M. Borden, E. Rank, and T. Hughes. An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces. *Comp. Meth. Appl. Mech. Engrg.*, 249 – 252:116 – 150, 2012.

[55] D. Schillinger, J. Evans, A. Reali, M. Scott, and T. Hughes. Isogeometric collocation: Cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations. *Comp. Meth. Appl. Mech. Engrg.*, 267:170–232, 2013.

[56] D. Schillinger, S. Hossain, and T. Hughes. Reduced Bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis. *Comp. Meth. Appl. Mech. Engrg.*, 277(0):1 – 45, 2014.

[57] L. Schumaker. *Spline Functions: Basic Theory*. Cambridge University Press, third edition, 2007.

[58] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

[59] C. F. van Loan and N. Pitsianis. Approximation with Kronecker products. In *Linear algebra for large scale and real-time applications (Leuven, 1992)*, volume 232 of *NATO Adv. Sci. Inst. Ser. E Appl. Sci.*, pages 293–314. Dordrecht, 1993.